UNICOS® User Commands Reference Manual

SR–2011 10.0

Portions of this product may still be in development. The existence of those portions still in development is not a commitment of actual release or support by Cray Research, Inc. Cray Research, Inc. assumes no liability for any damages resulting from attempts to use any functionality or documentation not officially released and supported. If it is released, the final form and the time of official release and start of support is at the discretion of Cray Research, Inc.

New Features

The *UNICOS User Commands Reference Manual*, Cray Research publication SR–2011, incorporates the following changes for the UNICOS 10.0 release. These changes are new since the UNICOS 9.0 release.

Command	Change
`ascheck`	New for array session modification of array services.
`chacid`	Allows users to set the account IDs of symbolic links as well as regular files. A new `-h` option allows requested operations on symbolic links.
`limit`	New `-d` option enables users to disable core file creation.
`mvf`	New tape filter that handles input, output, and volume switches.
`netstat`	New options: • `-A` causes kernel memory address of route data structures to be printed • `-Ar` prints kernel memory addresses • `-k` forces the command to use `/usr/kmem` to obtain data rather than system calls
`quota`, `quotamon`	Updated for optional aggregate quota feature.
`rlogin`	Two new options: `-E`, which stops characters from being removed as an escape character, and `-L`, which allows `rlogin` to be run in `-opost` mode.
`rsv`	New `-t` option specifies placement of message file.
`sm`	New `-o` option used to specify backward compatible PL types for USM.
`target`	New `-s` option prints only the machine subtype field.
`tplist`	New `-a` option verifies the output from a copy operation.

`tpmnt`	Enhancements to override special meanings for 1998 and 1999 expiration dates.
`tpquery`	New command allows you to query an autoloader server program to find out whether one or more volume serial numbers are within the domain of the queried autoloader.
`tpstat`	New `-a` option outputs device status from all tape devices, except those with `DOWN` status. Also, specification of over-committed mount requests is now permitted.

Record of Revision

Version	*Description*
1.0	March 1986. Documentation supporting the UNICOS 1.0 release running on Cray Research computer systems.
1.1	July 1986. Online documentation supporting the UNICOS 1.0 release running on Cray Research computer systems.
2.0	October 1986. Documentation supporting the UNICOS 2.0 release running on Cray Research computer systems.
3.0	July 1987. Documentation supporting the UNICOS 3.0 release running on Cray Research computer systems.
4.0	July 1988. Documentation supporting the UNICOS 4.0 release running on Cray Research computer systems.
5.0	March 1989. Documentation supporting the UNICOS 5.0 release running on Cray Research computer systems.
5.0	May 1990. This reprint to restock incorporates addendum pages from the *UNICOS 5.0 Release Addendum to Software Publications*, Cray Research publication SR-2096.
6.0	February 1991. Documentation supporting the UNICOS 6.0 release running on Cray Research computer systems.
7.0	September 1992. Documentation supporting the UNICOS 7.0 release running on Cray Research computer systems.
8.0	February 1994. Documentation supporting the UNICOS 8.0 release running on Cray Research computer systems.

9.0	September 1995. Documentation supporting the UNICOS 9.0 release running on Cray Research computer systems.
10.0	November 1997. Documentation supporting the UNICOS 10.0 release running on Cray Research computer systems. See the New Features section for a description of the new material in this release.

Preface

This publication documents UNICOS release 10.0 running on the following Cray Research systems:

- Systems with an IOS model V (IOS-V)
  - CRAY J90 series
- Systems with an IOS model E (IOS-E)
  - CRAY T90 series with Cray floating-point CPUs
  - CRAY T90 series with IEEE Std. 754 floating-point CPUs
  - CRAY C90 series
- Systems with a GigaRing IOS
  - CRAY T90 series with Cray floating-point CPUs
  - CRAY T90 series with IEEE Std. 754 floating-point CPUs
  - CRAY J90se series
- Cray Research hosts with an IOS-E for CRAY T3D systems
  - Cray C90 series

This publication also supplements the information contained in the other manuals in the UNICOS documentation set.

Related publications

The following man page manuals contain additional information that may be helpful.

Note: For the UNICOS 10.0 release, man page reference manuals may not be ordered in printed book form. Instead, they are available as printable PostScript files provided on the same DynaWeb CD as the rest of the supporting documents for this release. Individual man pages are still available online and can be accessed by using the `man`(1) command.

- *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

- *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014
- *UNICOS Administrator Commands Reference Manual*, Cray Research publication SR–2022
- *UNICOS System Libraries Reference Manual*, Cray Research publication SR–2080

The following ready references are available in printed form from the Distribution Center:

- *UNICOS User Commands Ready Reference*, Cray Research publication SQ–2056
- *UNICOS System Libraries Ready Reference*, Cray Research publication SQ–2147
- *UNICOS System Calls Ready Reference*, Cray Research publication SQ–2215
- *UNICOS Administrator Commands Ready Reference*, Cray Research publication SQ–2413

The following manuals provide additional information about the UNICOS operating system and related subjects.

Introductory manuals:

- *TCP/IP Network User's Guide*, Cray Research publication SG–2009
- *UNICOS Text Editors Primer*, Cray Research publication SG–2050
- *Tape Subsystem User's Guide*, Cray Research publication SG–2051

System administration manuals:

- *General UNICOS System Administration*, Cray Research publication SG–2301
- *I/O Subsystem (IOS) Operator's Guide for UNICOS*, Cray Research publication SG–2005

Reference manuals:

- *Application Programmer's Library Ready Reference*, Cray Research publication SQ–2231
- *CF90 Commands and Directives Reference Manual*, Cray Research publication SR–3901
- *CF90 Ready Reference*, Cray Research publication SQ–3900

- *Cray Assembly Language (CAL) for Cray PVP Systems Reference Manual*, Cray Research publication SR–3108
- *Cray Standard C Reference Manual*, Cray Research publication SR–2074
- *Remote Procedure Call (RPC) Reference Manual*, Cray Research publication SR–2089
- *UNICOS Shells Ready Reference*, Cray Research publication SQ–2116
- *UNICOS Environment Variables Ready Reference*, Cray Research publication SQ–2117
- *UNICOS vi Reference Card*, Cray Research publication SQ–2054
- *UNICOS ed Reference Card*, Cray Research publication SQ–2055

Library reference manuals:

- *Application Programmer's Library Reference Manual*, Cray Research publication SR–2165
- *Scientific Libraries Reference Manual*, Cray Research publication SR–2081
- *Intrinsic Procedures Reference Manual*, Cray Research publication SR–2138

Ordering Cray Research publications

The *User Publications Catalog*, Cray Research publication CP–0099, describes the availability and content of all Cray Research hardware and software documents that are available to customers. Cray Research customers who subscribe to the Cray Inform (CRInform) program can access this information on the CRInform system.

To order a document, either call the Distribution Center in Mendota Heights, Minnesota, at +1–612–683–5907, or send a facsimile of your request to fax number +1–612–452–0141. Cray Research employees may send electronic mail to `orderdsk` (UNIX system users).

Customers who subscribe to the CRInform program can order software release packages electronically by using the `Order Cray Software` option.

Customers outside of the United States and Canada should contact their local service organization for ordering and documentation information.

Conventions

The following conventions are used throughout this document:

Convention	Meaning
`command`	This fixed-space font denotes literal items such as commands, files, routines, path names, signals, messages, and programming language structures.
`manpage`(*x*)	Man page section identifiers appear in parentheses after man page names. The following list describes the identifiers: 1 User commands 1B User commands ported from BSD 2 System calls 3 Library routines, macros, and opdefs 4 Devices (special files) 4P Protocols 5 File formats 7 Miscellaneous topics 7D DWB-related information 8 Administrator commands Some internal routines (for example, the `_assign_asgcmd_info`() routine) do not have man pages associated with them.
variable	Italic typeface denotes variable entries and words or concepts being defined.
`user input`	This bold, fixed-space font denotes literal items that the user enters in interactive sessions. Output is shown in nonbold, fixed-space font.
[ ]	Brackets enclose optional portions of a command or directive line.

...	Ellipses indicate that a preceding element can be repeated.

The following machine naming conventions may be used throughout this document:

Term	Definition
Cray PVP systems	All configurations of Cray parallel vector processing (PVP) systems.
Cray MPP systems	All configurations of the CRAY T3D series. The UNICOS operating system is not supported on CRAY T3E systems. CRAY T3E systems run the UNICOS/mk operating system.
All Cray Research systems	All configurations of Cray PVP and Cray MPP systems that support this release.

The default shell in the UNICOS and UNICOS/mk operating systems, referred to in Cray Research documentation as the *standard shell*, is a version of the Korn shell that conforms to the following standards:

- Institute of Electrical and Electronics Engineers (IEEE) Portable Operating System Interface (POSIX) Standard 1003.2–1992
- X/Open Portability Guide, Issue 4 (XPG4)

The UNICOS and UNICOS/mk operating systems also support the optional use of the C shell.

Cray UNICOS Version 10.0 is an X/Open Base 95 branded product.

Man page sections

The entries in this document are based on a common format. The following list shows the order of sections in an entry and describes each section. Most entries contain only a subset of these sections.

Section heading	Description
NAME	Specifies the name of the entry and briefly states its function.
SYNOPSIS	Presents the syntax of the entry.

IMPLEMENTATION	Identifies the Cray Research systems to which the entry applies.
STANDARDS	Provides information about the portability of a utility or routine.
DESCRIPTION	Discusses the entry in detail.
NOTES	Presents items of particular importance.
CAUTIONS	Describes actions that can destroy data or produce undesired results.
WARNINGS	Describes actions that can harm people, equipment, or system software.
ENVIRONMENT VARIABLES	Describes predefined shell variables that determine some characteristics of the shell or that affect the behavior of some programs, commands, or utilities.
RETURN VALUES	Describes possible return values that indicate a library or system call executed successfully, or identifies the error condition under which it failed.
EXIT STATUS	Describes possible exit status values that indicate whether the command or utility executed successfully.
MESSAGES	Describes informational, diagnostic, and error messages that may appear. Self-explanatory messages are not listed.
ERRORS	Documents error codes. Applies only to system calls.
FORTRAN EXTENSIONS	Describes how to call a system call from Fortran. Applies only to system calls.
BUGS	Indicates known bugs and deficiencies.
EXAMPLES	Shows examples of usage.
FILES	Lists files that are either part of the entry or are related to it.

SEE ALSO	Lists entries and publications that contain related information.

Reader comments

If you have comments about the technical accuracy, content, or organization of this document, please tell us. You can contact us in any of the following ways:

- Send us electronic mail at the following address:

  `publications@cray.com`

- Contact your customer service representative and ask that an SPR or PV be filed. If filing an SPR, use `PUBLICATIONS` for the group name, `PUBS` for the command, and `NO-LICENSE` for the release name.

- Call our Software Publications Group in Eagan, Minnesota, through the Customer Service Call Center, using either of the following numbers:

  1–800–950–2729 (toll free from the United States and Canada)

  +1–612–683–5600

- Send a facsimile of your comments to the attention of "Software Publications Group" in Eagan, Minnesota, at fax number +1–612–683–5599.

We value your comments and will respond to them promptly.

CONTENTS

NAME

`intro` – Introduces user utilities and commands

IMPLEMENTATION

All Cray Research systems

STANDARDS

The manual entry for each utility includes a section that lists the standard or standards in which the utility being described is defined. Do not infer, however, from the reference to a standard that the entire UNICOS set of utilities has necessarily been validated to conform to that standard. Validation of conformance to these standards is an issue discussed in other Cray Research documents.

In this manual, the reference to a standard provides you with information about the portability of code using that utility. For example, if the entry for the utility states that the utility is defined in the XPG4 standard, you can expect a given utility to be found in any vendor's system that conforms to the XPG4 standard. On the other hand, if the entry for the utility states that it is a Cray Research extension, you cannot expect it to be found in other vendors' systems.

The specific meanings of the terms in the STANDARDS section are as follows:

Term	Description
XPG4	Defined in X/Open Company Ltd., Single UNIX Specification Spec 1170.
POSIX	Defined in POSIX IEEE Std 1003.2-1992, *Shell and Utilities*.
AT&T extension	Not defined in the POSIX standard; it originated from one or more of the software releases from AT&T.
BSD extension	Not defined in the POSIX standard; it originated from the Fourth Berkeley Software Distribution under license from The Regents of the University of California.
FSF extension	Not defined in the POSIX standard; it originated from the Free Software Foundation (FSF) under terms of the GNU General Public License.
CRI extension	Not defined in the POSIX standard; added by Cray Research.

DESCRIPTION

This manual presents, in alphabetical order, the UNICOS user utilities and commands for all Cray Research systems.

Command Syntax Terminology

The following terms identify the components of a UNICOS utility:

Utility Component	Definition
Command	The name of an executable file in lowercase letters.

Option	A command-line element indicated by a hyphen and usually followed by a single lowercase letter.
Option-argument	A character string supplying information for the preceding option. Although there are some exceptions, option-arguments are usually not optional.
Operand	A command-line element to be passed to the utility; not associated with an option. Operands can be optional.

Items enclosed in square brackets, [ ], are optional. *White space* refers to any number of horizontal spaces or tabs.

For a more detailed description of conventions, see *UNICOS Command Conventions*, Cray Research publication CP–2058.

EXIT STATUS

On termination, each command returns 2 bytes of status; one supplied by the system and giving the cause for termination, and (in the case of "normal" termination) one supplied by the procedure (see `wait`(2) and `exit`(2)). The former byte is 0, indicating normal termination; the latter is usually 0, indicating successful execution, and nonzero indicating troubles such as erroneous parameters, bad or inaccessible data, or other inability to cope with the task at hand. It is called variously exit code, exit status, or return status, and is described only where special conventions are involved.

BUGS

Many commands do not use the aforementioned syntax.

SEE ALSO

`getopt`(1), `getopts`(1) in the *UNICOS User Commands Reference Manual*, Cray Research publication SR–2011

`getopt`(3C) in the *UNICOS System Libraries Reference Manual*, Cray Research publication SR–2080

UNICOS Command Conventions, Cray Research publication CP–2058

NAME

acctcom – Searches and prints process accounting files

SYNOPSIS

acctcom [-a] [-b] [-c] [-d] [-f] [-h] [-i] [-k] [-m] [-p] [-q] [-r] [-t] [-v] [-w] [-y] [-A] [-B] [-D] [-F] [-J] [-M] [-N] [-P] [-T] [-U] [-V] [-W] [-X] [-Y] [-Z] [-e *time*] [-g *group*] [-j *jid*] [-l *line*] [-n *pattern*] [-o *ofile*] [-s *time*] [-u *user*] [-C *sec*] [-E *time*] [-H *factor*] [-I *chars*] [-O *sec*] [-S *time*] [*files*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The acctcom utility reads data *files*, in the format described by acct(5), and writes selected records to standard output. You can specify *files* to be read; otherwise, the standard input or /usr/adm/acct/day/pacct is read. Each record represents the execution of one process.

The acctcom utility accepts three types of options: output file options, selection options, and printing options.

Output Options

-o *ofile* Copies selected process records in the input data format to *ofile*, an output file you specify. Suppresses standard output printing.

Selection Options

-e *time* Selects processes existing at or before *time*, given in the format [D*dd*:]*hh*[:*mm*[:*ss*]]. (See EXAMPLES.) The letter D flags the presence of the relative day offset parameter, which allows acctcom to select records from a previous day. (To determine the day offset, use the -W option.)

-g *group* Selects only processes belonging to *group*. You can designate the *group* by either the group ID or group name.

-j *jid* Selects only processes that have a job ID that matches the *jid* argument.

-l *line* Selects only processes that belong to terminal /dev/*line*.

-n *pattern* Selects only commands matching *pattern* that may be a regular expression, as in ed(1), except that a + symbol indicates one or more occurrences.

-s *time* Selects processes existing at or after *time*, given in the format [D*dd*:]*hh*[:*mm*[:*ss*]]. See the -e option and EXAMPLES for more information on this format. Using the same *time* for both -s and -e shows the processes that existed at *time*.

`-u` *user* Selects only processes that belong to *user*. May be specified by a user ID, a login name that is then converted to a user ID, a # symbol designating only those processes executed with super-user privileges, or a question mark (`?`) designating only those processes associated with an unknown user ID. To avoid interpretation by the shell, the question mark must be escaped.

`-C` *sec* Selects only processes with total CPU time (system plus user time) exceeding *sec* seconds.

`-E` *time* Selects processes ending at or before *time*, given in the format [`D`*dd*`:`]*hh*[`:`*mm*[`:`*ss*]]. See the `-e` option and EXAMPLES for more information on this format.

`-H` *factor* Selects only processes that exceed *factor*. Factor is the "hog factor" (as explained in the description of printing option `-h`).

`-I` *chars* Selects only processes that transfer more characters than the cutoff number given by *chars*.

`-O` *sec* Selects only processes with CPU system time exceeding *sec* seconds.

`-S` *time* Selects processes that start at or after *time*, given in the format [`D`*dd*`:`]*hh*[`:`*mm*[`:`*ss*]]. See the `-e` option for more information on this format.

Printing Options

`-a` Shows some average statistics about the processes selected. The statistics are printed after the output records.

`-b` Reads backward, showing latest commands first. This option has no effect when the standard input is read.

`-c` Prints additional I/O counts for buffered and raw blocks transferred.

`-d` Prints the number of clock periods past the last second mark to the actual start time.

`-f` Prints the `fork/exec` flag in octal and system exit status in decimal in the output. The lower 8 bits from the exit status of the process are reported in the exit status (`STAT`) column. For more information about these bits, see `wait`(2).

`-h` Prints the fraction of total available CPU time, instead of mean memory size, consumed by the process during its execution. This is known as the "hog factor" and is computed as follows:

(total CPU time)/(elapsed time)

`-i` Prints columns that contain the I/O counts in the output.

`-k` Prints total kcore-minutes instead of memory size. This is an integral of memory usage over time. One kcore-minute is 1024 words used for 1 minute.

`-m` Prints mean core size. This is the default print option. If you do not specify any other print options, `-m` is selected. If do specify other print options and you want mean core size to print, you must specify `-m`.

`-p` Prints process ID and parent process ID.

`-q` Prints only the average statistics as with the `-a` option. Does not print any output records.

`-r` Prints the CPU factor (user time/(system time + user time)).

`-t` Prints separate system and user CPU times.

`-v` Excludes column headings from the output.

`-w` Prints wait times.

`-y` Prints total system call time.

`-A` Print a column with account IDs.

`-B` Prints breakdown of multitasking time. If the default Hardware Performance Monitor (HPM) group is changed to anything but 1, the `WAIT SEMA` field will always be 0. This is because wait semaphore time is accumulated by HPM counter group 1.

`-D` Prints only device accounting information. This option must not be used with other print options.

`-F` Prints secondary data segments (SDS) usage information.

`-J` Prints a column with job IDs.

`-M` Prints additional memory usage data.

`-N` Prints `nice` field value, or scheduling priority for use of CPU time. The range of values typically is 1 through 40, with 1 being super user and higher `nice` values being lower priorities.

`-P` Prints Cray MPP usage information where a CRAY T3D system is attached.

`-T` Prints only end-of-job termination data. This option must not be used with other print options.

`-U` Shows all user IDs in numeric format.

`-V` Combines some I/O fields; must be specified with the `-c`, `-w`, or `-D` option.

`-W` Prints the start and end dates and each date change found in the file. Ignores all other print and selection options. This option is useful if your data spans more than 1 day (that is, not 00:00 to 00:00) and if more than 1 day of data is present in the `pacct` file. The day number of the date change is printed and can be used with the time selects. (See Example 2.)

`-X` Prints the process start date. The date follows the last date printed on the line and will be in the format: *day month date year* (for example, `Sun Sep 16 1993`).

`-Y` Prints the process end date. The date follows the last data printed on the line and will be in the format: *day month date year* (for example, `Sun Sep 16 1993`).

`-Z` Skips (does not print) first seven fields (must be specified with one of the print options (`-cdfhikmprtwyABDFJMNT`)).

files Input file(s) you specify. These are one or more of the `/usr/adm/acct/day/pacct*` files. If you do not specify *files*, and if the standard input is associated with a terminal or `/dev/null` (as is the case when using & in the shell), `/usr/adm/acct/day/pacct` is read; otherwise, the standard input is read. Therefore, when executing `acctcom` using the Network Queuing System (NQS), you must specify *files*.

Any file arguments specified are read in their respective order. Each file is usually read forward; that is, in chronological order by process completion time. The `/usr/adm/acct/day/pacct` file is the current file to be examined.

The output fields are as follows:

Field Name	Option	Definition
`COMMAND NAME`	`! Z`	ASCII command name OR #command name if executed with super-user privileges
`USER`	`! Z`	ASCII user name
`TTYNAME`	`! Z`	ASCII tty name (`?` in this field indicates that a process is not associated with a known terminal)
`START TIME`	`! Z`	Start time of process in clock format (that is, 10:01:25)
`END TIME`	`! Z`	End time of process in clock format (that is, 10:10:15)
`REAL (SECS)`	`! Z`	Elapsed time of process in seconds
`CPU (SECS)`	`! t`	CPU time used by the process in seconds OR
`(SECS) SYS`	`-t`	System time used by the process in seconds
`(SECS) USER`	`-t`	User time used by the process in seconds
`CHARS TRNSFD`	`-i`	Number of characters transferred
`PHYS REQS`	`-i`	Number of physical IO requests
`CPU FACTOR`	`-r`	User time divided by the CPU time
`HOG FACTOR`	`-h`	CPU time divided by the elapsed time
`MEAN SIZE(K)`	`-m`	Average amount of memory used by the process in kilowords
`KCORE MIN`	`-k`	Amount of memory used by the process in kilowords * minutes
`F STAT`	`-f`	The record flags, `F` (the `fork/exec` flag: `1` for `fork` without `exec`), and exit status
`JOB ID`	`-J`	Job identifier
`ACCT ID`	`-A`	Account identifier
`GRP ID`	`-G`	Group identifier
`LOGIO REQS`	`-c`	Number of logical IO requests
`PHYS BLKS`	`-cV`	Number of physical blocks transferred OR
`PHYS MVD: BUF`	`-c`	Number of buffered physical blocks transferred

Field Name	Option	Definition
PHYS MVD: RAW	-c	Number of raw physical blocks transferred
PID	-p	Process identifier
PPID	-p	Parent process identifier
START FRACT	-d	Fractional second part of the start time (.*xx*)
IOWAIT COMB	-wV	Combined I/O wait time in seconds OR
IOWAIT LOCKED	-w	Locked in memory I/O wait time in seconds
IOWAIT UNLOCK	-w	Unlocked in memory I/O wait time in seconds
IOWAIT TERM	-w	Total I/O wait time in seconds
WAIT SWAP	-w	I/O wait time in seconds
SCTIME SECS	-y	System call time in seconds
SDS MWSECS	-F	SDS memory integral in megawords * seconds
SDS REQS	-F	Number of SDS logical requests
SDS CHARS	-F	Number of SDS characters transferred
MPP PE'S	-P	Number of Cray MPP process elements used
MPP BB'S	-P	Number of Cray MPP barrier bits used
MPP (SECS)	-P	Time the Cray MPP was used in seconds
WAIT SEMA	-B	Time spent waiting for semaphores in seconds
USERTM CPU	-B	User time each CPU executed in seconds
IOWAIT LCKMEM	-M	I/O wait memory divided by I/O wait time in kilowords
HIMEM	-M	Largest amount of memory in kilowords
SWAPS	-M	Number of I/O swaps
NICE	-N	Nice value of the process
PROCESS START DATE	-W	Date that the process started in `ctime` format
PROCESS END DATE	-W	Date that the process ended in `ctime` format

NOTES

The `-d` option no longer displays the system run queue time.

BUGS

The `acctcom` utility reports only on processes that have terminated; use `ps`(1) for active processes.

If *time* exceeds the present time, *time* is interpreted as occurring on the previous day.

EXAMPLES

Example 1: The following example generates a list of commands executed by user `samuel` by examining all current process accounting files. The output includes system and user CPU times. In this example, if the `pacct` files are not specified in the order shown, the commands may not be reported in ascending time order.

```
acctcom -u samuel -t /usr/adm/acct/day/pacct?* /usr/adm/acct/day/pacct
```

Example 2: The following example shows how, using the printing option `-W`, the day number of the date change is printed.

```
acctcom -W

Day  0:  Mon Apr  3 10:20:11 1995 - first record
Day  1:  Tue Apr  4 00:00:00 1995 - date change
Day  4:  Fri Apr  7 10:20:00 1995 - date change
Day  4:  Fri Apr  7 14:43:10 1995 - last record
```

Example 3: The following example shows how you would select and print data from day 4, 10:20 A.M. in Example 2; you would use the same format to specify dates and times when using selection options `-e`, `-E`, `-s`, `-S`.

```
acctcom -s D4:10:20
```

Example 4: The following example shows how you would select and print data from the `pacct` file for an interval on day 4 between 8:00 A.M. and 4:00 P.M.

```
acctcom -S D4:08:00:00 -E D4:16:00:00
```

FILES

`/etc/group`	Group file that contains group names and group IDs
`/etc/udb`	User validation file that contains user control limits
`/usr/adm/acct/day/pacct`	Process accounting file that contains resource usage information for processes running on the system

SEE ALSO

`ps`(1), `su`(1)

`acct`(2), `wait`(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

`acct`(5), `utmp`(5) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

`acct`(8), `acctcms`(8), `acctcon`(8), `acctmerg`(8), `acctprc`(8), `acctsh`(8), `fwtmp`(8), `runacct`(8) in the *UNICOS Administrator Commands Reference Manual*, Cray Research publication SR–2022

NAME

adb – Invokes the absolute debugger

SYNOPSIS

adb [-w] [*objfile* [*corefile*]]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The adb utility invokes the absolute debugger, which lets you look at core files resulting from crashed systems or aborted programs. adb is an interactive, general-purpose debugger with access to global symbols. It lets you display output in a variety of formats, patch files, and run programs with embedded breakpoints.

The adb utility accepts the following options and operands:

-w Indicates that both *objfile* and *corefile* should be created if necessary and opened for reading and writing so that files can be modified using adb requests.

objfile An executable program file, preferably containing a symbol table. If it does not contain a symbol table, the symbolic features of adb cannot be used, although you can still examine the file. The default for *objfile* is a.out.

corefile A core image file produced after executing *objfile*; the default for *corefile* is core. See core(5) for a description of core files.

Generally, requests to adb are in the following format:

[*address*] [, *count*] [*command modifier*]

address and *count* are expressions (see the Expressions subsection). adb maintains a current address called *dot*. If *address* is present, *dot* is set to *address*; otherwise, *dot* refers to the address of the last item printed. For example, the following request sets *dot* to octal 100 and prints the instruction at that address:

```
0100?i
```

The following request prints 15 octal numbers starting at address *dot*:

```
.,15/o
```

Initially *dot* is set to 0. The interpretation of an address depends on the context in which it is used. If a subprocess is being debugged, addresses are interpreted in the usual way in the address space of the subprocess. For further details of address mapping, see the Addresses subsection.

For most requests, *count* specifies how many times the request is executed; default *count* is 1.

The `adb` utility ignores `QUIT`, and `INTERRUPT` causes `adb` to return to the next `adb` request. Use the `$Q`, `$q`, or `<CONTROL-d>` keys to exit from `adb`.

Expressions

Addresses are represented by expressions. Expressions consist of decimal, octal, and hexadecimal integers, and symbols from the program being tested. Integers are assumed to be octal by default. Expressions consist of the following characters:

`.` Value of *dot*.

`+` Value of *dot* incremented by the current increment.

`^` Value of *dot* decremented by the current increment.

`"` Last *address* typed.

integer An octal number. If you have used a `$d` request, *integer* is a decimal number unless the number begins with a leading `0x` (for hexadecimal) or a leading `0` (for octal).

integer`.`*fraction*
A 64-bit floating-point octal number.

'cccccccc' The ASCII value of up to 8 characters. A backslash (\) may be used to escape a `'` symbol.

`<` *name* Value of *name*, which is either a variable name or a register name. `adb` maintains a number of variables (see the Variables subsection) named by single uppercase letters or digits. If *name* is a register name, the register value is obtained from the user structure in *corefile*. The register names are `a0` through `a7`, `s0` through `s7`, `p`, `vl`, `vm`, `v000` through `v777`, `b00` through `b77`, `t00` through `t77`, `sb0` through `sb7`, `sm00` through `sm37`, and `st0` through `st7`.

symbol Sequence of uppercase or lowercase letters, underscores or digits, not starting with a digit. A backslash (\) may be used to escape other characters. The value of *symbol* is taken from the symbol table in *objfile*.

(*exp*) Value of the expression *exp*.

You can also use monadic and dyadic operators. Dyadic operators are left-associative and are less binding than monadic operators.

`*`*exp* Contents of the location addressed by *exp* in *corefile*

`@`*exp* Contents of the location addressed by *exp* in *objfile*

`-`*exp* Integer negation

~ exp Bitwise complement

exp1`+`*exp2* Integer addition

exp1`-`*exp2* Integer subtraction

exp1`*`*exp2*	Integer multiplication
exp1`%`*exp2*	Integer division
exp1`&`*exp2*	Bitwise conjunction (and)
exp1`|`*exp2*	Bitwise disjunction (or)
exp1`#`*exp2*	*exp1* rounded up to the next multiple of *exp2*

Commands

Most commands to `adb` consist of a verb, followed by a modifier or list of modifiers. The following verbs are available. (The verbs `?` and `/` may be followed by `*`; see the Addresses subsection for further details.)

Verb	Description
`?`	Prints the contents from *objfile*
`/`	Prints the contents from *corefile*
`=`	Prints the current value of *dot*
`$`	Designates a miscellaneous request
`:`	Manages a child process
`!`	Creates a shell to read the rest of the line following `!`
`>`	Designates an assignment request

The following list shows the combinations of requests and modifiers you can issue to `adb`. Use `?` with the [`?`/]`l`, [`?/`]`w`, and [`?/`]`m` requests if you want to include the contents from *objfile*; use `/` if you want to include the contents from *corefile*.

`newline` Repeats the previous request with a *count* of 1 (works only with the ? and / requests).

[`?/`]`l` *value mask*
: Masks words starting at *dot* with *mask* and compares them with *value* until *count* matches are found. If `L` is used, the match is for 8 bytes at a time instead of 2. If no match is found, *dot* is unchanged; otherwise, *dot* is set to the matched location. If *mask* is omitted, –1 is used.

[`?/`]`w` *value* ...
: Writes the 2-byte *value* into the addressed location. If the request is `W`, it writes 8 bytes. Word-aligned addresses are required when writing to a subprocess address space.

[`?/`]`m` *b1 exp1 f1*[`?/`]
: Records new values for (*b1*, *exp1*, and *f1*). If less than three expressions are given, the remaining map parameters are left unchanged. If the `?` or `/` is followed by `*`, the second segment (*b2*, *exp2*, and *f2*) of the mapping is changed. If the list is terminated by `?` or `/`, the file (*objfile* or *corefile*, respectively) is used for subsequent requests. (For example, `/m?` causes `/` to refer to *objfile*.)

`>name` Assigns *dot* to the variable or register specified.

`$`*modifier* Specifies miscellaneous requests. The available *modifier*s are as follows:

`<`*file* Reads commands from the file *file* and returns.

`>`*file* Sends output to the file *file*, which is created if it does not exist. If no file name is given, returns to `stdout`.

`r` Prints the general registers, prints the instruction addressed by pc, and sets *dot* to pc. `b00` is also displayed.

`b` Prints all breakpoints and their associated counts and commands.

`c` c stack backtrace; attempts a backtrace for all languages. If *address* is given, it is taken as the address of the current frame. If *count* is given, only the first *count* frames are printed.

`w` Sets the page width for output to *address* (default 80).

`s` Sets the limit for symbol matches to *address* (default 4096 bytes).

`o` Specifies that all integers input are regarded as octal. This is the default.

`d` Switches input integers as specified. To switch from

octal to decimal, enter:	`012$d`
octal to hexadecimal, enter:	`020$d`
hexadecimal to decimal, enter:	`0xa$d`
decimal to hexadecimal, enter:	`16$d`

`p` Changes the register set address. To change from one register set to another, *n*`$p` moves the register and local memory map so that register set *n* may be examined. The numbering starts from 0. Thus, $0 \leq n < var[VARQ]$.

`q` Exits from `adb`.

`u` Prints all nonzero variables in octal.

`m` Prints the address map.

`v` Prints vector registers at *address* for *count*.

`j` Prints `b` registers at *address* for *count*.

`k` Prints `t` registers at *address* for *count*.

`:`*modifier* Manages a subprocess. Available *modifier*s are as follows:

`b`*c* Sets breakpoint at *address*. The breakpoint is executed *count*–1 times before causing a stop. Each time the breakpoint is encountered, the request *c* is executed.

`d` Deletes breakpoint at *address*.

`r` Runs *objfile* as a subprocess. If *address* is given explicitly, the program is entered at this point; otherwise, the program is entered at its standard entry point. The value *count* specifies how many breakpoints are to be ignored before stopping. Arguments to the subprocess may be supplied on the same line as the request. An argument starting with `<` or `>` causes the standard input or output to be established for the request. All signals are enabled on entry to the subprocess.

`c`*s* Continues the subprocess with signal *s* (see `signal`(2)). If *address* is given, the subprocess is continued at this address. If no signal is specified, the signal that caused the subprocess to stop is sent. Breakpoint skipping is the same as for `r`.

`s`*s* Continues the subprocess the same as for `c`, except that the subprocess is single stepped *count* times. If there is no current subprocess, *objfile* is run as a subprocess as for `r`. In this case, no signal can be sent; the remainder of the line is treated as arguments to the subprocess.

`k` Terminates the current subprocess, if any.

Formats

A *format* consists of one or more characters that describe the format of the output. Each format character may be preceded by an integer and an asterisk, for example, indicating a repeat count for the format character (3*x). While parsing a format, *dot* is incremented by the amount given for each format letter. If no format is given, the previous format is used. The format letters available are as follows:

`o` 8 Prints 8 bytes in octal. All octal numbers output by `adb` are preceded by 0.
`O` 8 Prints 8 bytes in octal.
`q` 8 Prints in signed octal.
`Q` 8 Prints long signed octal.
`d` 8 Prints in decimal.
`D` 8 Prints long decimal.
`x` 8 Prints 8 bytes in hexadecimal.
`X` 8 Prints 8 bytes in hexadecimal.
`u` 8 Prints as an unsigned decimal number.
`U` 8 Prints long unsigned decimal.
`b` 1 Prints the addressed byte in octal. Numeric formats:
- 1 Same as `b`
- 2 Prints 2-byte parcels in octal
- 4 Prints 4-byte halfwords in octal

`c` 1 Prints the addressed character.
`C` 1 Prints the addressed character using the following escape convention. Character values 000 through 040 are printed as @ followed by the corresponding character in the range 0100 through 0140. The character @ is printed as @@.
`s` *n* Prints the addressed characters until a 0 character is reached.
`S` *n* Prints a string by using the @ escape convention; *n* is the length of the string including its zero terminator.
`Y` 8 Prints 4 bytes in date format (see `ctime`(3C)).
`i` *n* Prints as instructions; *n* is the length of instructions.
`a` 0 Prints the value of *dot* in symbolic form.
`p` 8 Prints the addressed value in symbolic form, using the same rules for symbol lookup as `a`.
`t` 0 When preceded by an integer, it tabs to the next appropriate tab stop. For example, `8t` moves to the next 8-space tab stop.
`r` 0 Prints a space.
`n` 0 Prints a new line.
`"..."` 0 Prints the enclosed string.
`^` *dot* decrements by 1 word; nothing is printed.

+ *dot* is incremented by 1; nothing is printed.
- *dot* is decremented by 1; nothing is printed.
space Increments *dot* by one word.

Variables

The `adb` utility provides several uppercase variables. Named variables are set initially by `adb`, but they are not subsequently used. You can access locations by using the `adb`-defined variables. The `adb` program reads the header of the core image file to find the values for these variables. If the second file specified does not seem to be a core file, or if it is missing, the header of the executable file is used instead.

`B` Base address of the data segment
`D` Data segment size
`E` Entry point
`M` "Magic" number (0405, 0407, 0410, or 0411)
`S` Stack segment size
`T` Text segment size
`P` Number of the current register set
`Q` Total number of register sets in the `core` file

Numbered variables are reserved for communication, as follows:

`0` Last value printed
`1` Last offset part of an instruction source
`2` Previous value of variable `1`

Addresses

All addresses are byte addresses, except for the register number in a `$v` display. The interpretation of an address depends on its context. If a subprocess is being debugged, addresses are interpreted in the usual way (as described below) in the address space of the subprocess.

For the UNICOS operating system, the second map for the `core` file is the map for the data section if there is split code and data.

If either file is not of the kind expected, *b1* and *f1* are set to 0 and *exp1* is set to the maximum file size; in this way, the whole file can be analyzed with no address translation.

EXIT STATUS

Exit status is 0, unless last command failed or returned nonzero status.

EXAMPLES

The following example shows a run of an adb debugger on a simple C program, which examines the symbol table at the end of an executable file. User input is shown in bold.

```
$ cat junk.c
# include <stdio.h>
main(){ printf("hello world\n");}
$ ./a.out/
hello world
$ adb  - a.out
$m
? map    ‘-’
b1 = 0   e1       = 40000000000000000f1 =  0
b2 = 0   e        = 0                     f2 = 0
/ map    ‘a.out’
b1 = 0   e1       = 326220                f1 = 70000
b2 = 0   e2       = 326220                f2 = 70000
/m 0 050000000000 0
0,4/00na
0:     0411    021547
20:    014624  016253
40:    03217   0
60:    0       01
100:
0100+0215470+0146240,30/O^tt010*Cna
364030:    0270000000040000003217   .@‘@‘@a@‘@‘@f@o
364040:   0141000001400000000001   B@‘@‘@@@‘@‘@‘@a
364050:                   03334200   @‘@‘@‘@‘@‘@m8@‘
364060:    0221012525410124400000   $AUXAR@‘@‘
364070:   0142400001440000000001   E@‘@‘H@‘@‘@‘@a
364100:                   03641500   @‘@‘@‘@‘@‘@oC@@
364110:    0221062224751723400000   $FIOON@‘@‘
364120:   0142400002000000000001   E@‘@a@‘@‘@‘@‘@a
364130:                   03641600   @‘@‘@‘@‘@‘@oC@‘
364140:    0221322404252221430503   $ZPERF1C
364150:   0141000002040000000001   B@‘@a@h@‘@‘@‘@a
364160:                   03334300   @‘@‘@‘@‘@‘@m8@@
364170:    0221423106656335060564   $bdmstat
364200:   0141000002040000000001   B@‘@a@h@‘@‘@‘@a
364210:                   03334400   @‘@‘@‘@‘@‘@m9@‘
364220:    0221463406456335060564   $fpistat
364230:   0141000002040000000001   B@‘@a@h@‘@‘@‘@a
364240:                   03334500   @‘@‘@‘@‘@‘@m9@@
364250:    0221573446456335060564   $oristat
```

```
364260:   0141000001640000000200   B@‘@‘h@‘@‘@‘@‘
364270:                   03334600   @‘@‘@‘@‘@‘@m9@‘
364300:    022164302711473l272000   $target@‘
364310:   0141000001640000000001   B@‘@‘h@‘@‘@‘@a
364320:                   03354600   @‘@‘@‘@‘@‘@mY@‘
364330:
$q
$ #
$ # run the program
$ ./a.out
hello world
$ # debug program, fixing invalid output
$ adb a.out -
write:b
:r
a.out: running
breakpoint       write:  s0  021564,0   $sysc$trap
(<a6+1)*010,2/OOna
414030:                          01                      044703
414050:                          014                     041364
414070:
041050/W 017
414050:                          14=                     17
0447030/XX
447030:                          68656c6c6f20776f        726c640a00000000
0447030/W 0x736f206c6f6e6720
447030:                          641453306615710073557=  715571006615733463440
0447040/W 0x7375636b65720a00
447040:                          7115431005000000000000=  715653066554534405000
:c
a.out: running
so long sucker
delbp: No such process
process                          terminated
$q
```

FILES

a.out	Executable file
core	Program memory dump

SEE ALSO

`ptrace`(2), `signal`(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

`ctime`(3C) in the *UNICOS System Libraries Reference Manual*, Cray Research publication SR–2080

`a.out`(5), `core`(5) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

NAME

addbss – Increases the amount of BSS space in an executable file

SYNOPSIS

addbss *file* [[*newfile*] *incr*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The addbss utility reports or changes the amount of BSS space for a program. The additional BSS space starts as free space in the memory manager.

The addbss utility accepts the following operands:

file Specifies an executable file.

newfile Specifies the new file, which contains *incr* more BSS space than *file*.

incr Specifies the amount of Kwords (1Kword is 1024 decimal) added to the BSS space of *file*.

If only one argument is present (the *file* argument), the amount of BSS space for that executable file is printed. *file* should represent an executable file. The format of an executable file is described in a.out(5). If two arguments are present (*file* and *incr*), *incr* Kwords are added to *file*'s BSS space. If three arguments are present (*file*, *newfile*, and *incr*), *file* is copied to *newfile* with a BSS size that is *incr* Kwords larger.

The *incr* argument must be at least 10.

NOTES

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
system, secadm	Allowed to increase BSS space for any executable file. In a privileged administrator shell environment, shell-redirected I/O is not subject to file protections.
sysadm	Allowed to increase BSS space for any executable file subject to security label restrictions. Shell-redirected I/O is subject to security label restrictions.

If the PRIV_SU configuration option is enabled, the super user is allowed to increase BSS space for any executable file. Shell-redirected I/O on behalf of the super user is not subject to file protections.

SEE ALSO

a.out(5) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

NAME

admin – Creates and administers SCCS files

SYNOPSIS

admin [-n] [-i[*name*]] [-r *rel*] [-t[*name*]] [-f *flag*[*flag-val*]] [-d *flag*[*flag-val*]] [-a *login*] [-e *login*] [-m *mrlist*] [-y[*comment*]] [-h] [-z] *files*

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The admin utility is used to create new Source Code Control System (SCCS) files and change parameters of existing ones. Arguments to admin, which may appear in any order, consist of keyletter arguments (keyletter arguments begin with the - symbol) and named files (SCCS file names must begin with the characters s.). If a named file does not exist, it is created, and its parameters are initialized according to the specified keyletter arguments. Parameters not initialized by a keyletter argument are assigned a default value. If a named file does exist, parameters corresponding to specified keyletter arguments are changed, and other parameters are left as is.

If a directory is named, admin behaves as though each file in the directory were specified as a named file, except that non-SCCS files (last component of the path name does not begin with s.) and unreadable files are silently ignored. If a name of - is given, the standard input is read; each line of the standard input is taken to be the name of an SCCS file to be processed. Again, non-SCCS files and unreadable files are silently ignored.

The keyletter arguments are as follows. Each is explained as though only one named file is to be processed because the effects of the arguments apply independently to each named file.

-n Indicates that a new SCCS file is to be created.

-i[*name*] Specifies the *name* of a file from which the text for a new SCCS file is to be taken. The text constitutes the first delta of the file (see -r keyletter for delta numbering scheme). If the -i keyletter is used, but the file name is omitted, the text is obtained by reading the standard input until an end-of-file is encountered. If this keyletter is omitted, the SCCS file is created empty. Only one SCCS file may be created by an admin utility on which the -i keyletter is supplied. Using a single admin to create two or more SCCS files requires that they be created empty (no -i keyletter). Note that the -i keyletter implies the -n keyletter.

`-r` *rel* Specifies the release into which the initial delta is inserted. This keyletter may be used only if the `-i` keyletter is also used. If the `-r` keyletter is not used, the initial delta is inserted into release 1. The level of the initial delta is always 1 (by default initial deltas are named 1.1).

`-t`[*name*] Specifies the *name* of a file from which descriptive text for the SCCS file is to be taken. If the `-t` keyletter is used and `admin` is creating a new SCCS file (the `-n` and/or `-i` keyletters are also used), the descriptive text file name must also be supplied. In the case of existing SCCS files: (1) a `-t` keyletter without a file name causes removal of any descriptive text currently in the SCCS file, and (2) a `-t` keyletter with a file name causes any text in the named file to replace any descriptive text currently in the SCCS file.

`-f` *flag* Specifies a *flag*, and possibly a value for the *flag*, to be placed in the SCCS file. Several `-f` keyletters may be supplied on a single `admin` command line. The allowable *flag*s and their values are:

`b` Allows use of the `-b` keyletter on a `get`(1) command to create branch deltas.

`c` *ceil* Specifies the highest release (that is, "ceiling") that may be retrieved by a `get`(1) command for editing. Must be a number greater than 0 but less than or equal to 9999. The default value for an unspecified `c` flag is 9999.

`f` *floor* The lowest release (that is, "floor") that may be retrieved by a `get`(1) command for editing. Must be a number greater than 0 but less than 9999. The default value for an unspecified `f` flag is 1.

`d` *SID* Specifies the default delta number (SID) to be used by a `get`(1) command.

`i` [*str*] Causes the `No id keywords (ge6)` message issued by `get`(1) or `delta`(1) to be treated as a fatal error. In the absence of this flag, the message is only a warning. The message is issued if no SCCS identification keywords (see `get`(1)) are found in the text retrieved or stored in the SCCS file. If a value is supplied, the keywords must exactly match the given string; however, the string must contain a keyword and no embedded new lines.

`j` Allows concurrent `get`(1) commands for editing on the same SID of an SCCS file. This allows multiple concurrent updates to the same version of the SCCS file.

`l` *list* Specifies a *list* of releases to that deltas can no longer be made (`get -e`) against one of these "locked" releases fails). The *list* has the following syntax:

list `::=` *range* `|` *list* `,` *range*
range `::=` *release number* `|` `a`

The character `a` in the *list* is equivalent to specifying all releases for the named SCCS file.

`n` Causes `delta`(1) to create a "null" delta in each of the releases skipped when a delta is made in a new release (for example, in making delta 5.1 after delta 2.7, releases 3 and 4 are skipped). These null deltas serve as anchor points so that branch deltas may be created from them later. The absence of this flag causes skipped releases to be nonexistent in the SCCS file, preventing branch deltas from being created from them in the future.

`q` *text* Specifies the user-definable text substituted for all occurrences of the `%Q%` keyword in SCCS file text retrieved by `get`(1).

`m` *mod* Specifies the module name of the SCCS file substituted for all occurrences of the `%M%` keyword in SCCS file text retrieved by `get`(1). If the `m` flag is not specified, the value assigned is the name of the SCCS file with the leading `s.` removed.

`t` *type* Specifies the *type* of module in the SCCS file substituted for all occurrences of `%Y%` keyword in SCCS file text retrieved by `get`(1).

`v` *pgm* Causes `delta`(1) to prompt for modification request (MR) numbers as the reason for creating a delta. The optional value specifies the name of an MR number validity checking program (see `delta`(1)). (If this flag is set when creating an SCCS file, the `-m` keyletter must also be used even if its value is null.)

`-d` *flag* Deletes the specified *flag* from an SCCS file. The `-d` keyletter may be specified only when processing existing SCCS files. Several `-d` keyletters may be supplied on a single `admin` command. See the `-f` keyletter for allowable *flag* names.

`-a` *login* Specifies a *login* name, or numerical UNIX system group ID, to be added to the list of users that may make deltas (changes) to the SCCS file. A group ID is equivalent to specifying all *login* names common to that group ID. Several `-a` keyletters may be used on a single `admin` command line. As many *login*s, or numerical group IDs, as desired may be on the list simultaneously. If the list of users is empty, then anyone may add deltas. If *login* or group ID is preceded by a `!` symbol, they are to be denied permission to make deltas.

`-e` *login* Specifies a *login* name, or numerical group ID, to be erased from the list of users allowed to make deltas (changes) to the SCCS file. Specifying a group ID is equivalent to specifying all *login* names common to that group ID. Several `-e` keyletters may be used on a single `admin` command line.

`-m` *mrlist* Inserts the list of modification requests (MR) numbers into the SCCS file as the reason for creating the initial delta in a manner identical to `delta`(1). The `v` flag must be set and the MR numbers are validated if the `v` flag has a value (the name of an MR number validation program). Diagnostics will occur if the `v` flag is not set or MR validation fails.

`-y`[*comment*] Inserts the *comment* text into the SCCS file as a comment for the initial delta in a manner identical to that of `delta`(1). Omission of the `-y` keyletter results in a default comment line being inserted in the following form:

```
date and time created
YY / MM / DD
HH : MM : SS
by
login
```

The `-y` keyletter is valid only if the `-i` and/or `-n` keyletters are specified (for example, a new SCCS file is being created).

`-h` Causes `admin` to check the structure of the SCCS file. (see `sccsfile`(5)), and to compare a newly computed checksum (the sum of all the characters in the SCCS file except those in the first line) with the checksum that is stored in the first line of the SCCS file. Appropriate error diagnostics are produced.

This keyletter inhibits writing on the file, so that it nullifies the effect of any other keyletters supplied, and is, therefore, only meaningful when processing existing files.

`-z` Recomputes the SCCS file checksum and stores it in the first line of the SCCS file (see keyletter `-h`).

NOTE: Use of this keyletter on a truly corrupted file may prevent future detection of the corruption.

files Specifies the files to be created or changed.

The last component of all SCCS file names must be of the form `s.`*file-name*. New SCCS files are given mode 444 (see `chmod`(1)). Write permission in the pertinent directory is, of course, required to create a file. All writing done by `admin` is to a temporary x-file, called `x.`*file-name*, (see `get`(1)), created with mode 444 if the `admin` utility is creating a new SCCS file, or with the same mode as the SCCS file if it exists. After successful execution of `admin`, the SCCS file is removed (if it exists), and the x-file is renamed with the name of the SCCS file. This ensures that changes are made to the SCCS file only if no errors occurred.

It is recommended that directories containing SCCS files be mode 755 and that SCCS files themselves be mode 444. The mode of the directories allows only the owner to modify SCCS files contained in the directories. The mode of the SCCS files prevents any modification at all except by SCCS commands.

If it should be necessary to patch an SCCS file, the mode may be changed to 644 by the owner, allowing use of `ed`(1). Care must be taken. The edited file should always be processed by an `admin -h` to check for corruption followed by an `admin -z` to generate a proper checksum. Another `admin -h` is recommended to ensure the SCCS file is valid.

The `admin` utility also makes use of a transient lock file (called `z.`*file-name*), which is used to prevent simultaneous updates to the SCCS file by different users. See `get`(1) for further information.

MESSAGES

Use help(1) for explanations.

EXAMPLES

The following example creates a new SCCS file named s.example.c. The file example.c is used for initial input and the SCCSID line is required for future deltas (see the EXAMPLES section of delta(1)).

```
$ cat example.c
static char SCCSID[] = "%Z%%M%      %I%   %G% %U%";
main()
{
        printf("Hello, world!\n");
}
$ admin -n -iexample.c -fi"`grep SCCSID example.c`" s.example.c
$
```

FILES

g-file	Existed before the execution of delta; removed after completion of delta.
p-file	Existed before the execution of delta; may exist after completion of delta.
q-file	Created during the execution of delta; removed after completion of delta.
x-file	Created during the execution of delta; renamed to SCCS file after completion of delta.
z-file	Created during the execution of delta; removed during the execution of delta.
d-file	Created during the execution of delta; removed after completion of delta.
/usr/bin/bdiff	Program to compute differences between the "gotten" file and the g-file.

SEE ALSO

cdc(1), comb(1), delta(1), ed(1), get(1), help(1), prs(1), rmdel(1), sact(1), sccs(1), sccsdiff(1), unget(1), val(1), vc(1), what(1)

chown(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

sccsfile(5) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

NAME

`airlogger` – Logs AIR messages in the system log

SYNOPSIS

`airlogger` [`-s` *msg_type*] [`-p` *product*] [`-P` *subproduct*] *message*

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `airlogger` utility logs automated incident reporting (AIR) messages in a system log file. The utility sends messages to the `airlog`(3C) library routine, which formats the messages and passes them to the `syslog`(3C) routine. `syslog` writes the messages in a UNICOS system log maintained by the `syslogd`(8) command.

The `airlogger` utility accepts the following options and operands:

`-s` *msg_type* Specifies the message type. The *msg_type* argument can be one of the following:

`start`	Indicates normal daemon initiation
`term`	Indicates normal daemon termination
`panic`	Indicates abnormal daemon termination
`crit`	Indicates that a disaster has occurred
`warn`	Contains (nonfatal) information that indicates possible future disaster
`atten`	Contains information that should be displayed for the operator
`info`	Contains useful information to be logged.
`pulse`	Contains a daemon heartbeat
`fork`	Indicates that the daemon created a child process
`user`	Contains user-entered text
`conf`	Contains configuration information

`-p` *product* Specifies the product to which the message pertains. The *product* can be one of the following:

`unicos`	UNICOS kernel
`nqs`	Network Queuing System (NQS)
`tcp`	TCP/IP
`tape`	Online tape subsystem
`dmf`	Data Migration Facility (DMF)
`nfs`	Network file system (NFS)
`acct`	System accounting
`disk`	CRI disks
`superl`	Open Systems Interconnection (OSI) product
`share`	Fair-share scheduler

`cron` `cron` daemon

`-P` *subproduct* Specifies information that further delineates the origin of the message. The *subproduct* argument is a comma-separated string. For example, if the *product* were `nqs`, a possible subproduct string could be `qfdaemon,readq,end`.

message Text to be logged.

The `airlog` library routine creates the format of the log entry by listing the arguments and adding an identifying key, the contents of which are defined based on the type argument. See the `airlog`(3C) man page for more information.

FILES

`/usr/logs/airlog` AIR system log file

SEE ALSO

`airlog`(3C), `syslog`(3C) in the *UNICOS System Libraries Reference Manual*, Cray Research publication SR–2080

`syslogd`(8) in the *UNICOS Administrator Commands Reference Manual*, Cray Research publication SR–2022

NAME

alias – Defines or displays aliases

SYNOPSIS

alias [-t] [-x] [*alias-name*[=*string*] ...]

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4
AT&T extensions (-t and -x options)

DESCRIPTION

The alias utility creates or redefines alias definitions or writes the values of existing alias definitions to standard output. An alias definition provides a string value that replaces a command name when it is encountered (see sh(1)).

An alias definition affects the current shell execution environment and the execution environments of the subshells of the current shell. The alias definition does not affect the parent process of the current shell or any utility environment that the shell invokes.

The alias utility supports the following options and operands:

-t	Sets or lists tracked aliases.
-x	Sets or lists exported aliases.
alias-name	Writes the alias definition to standard output.
alias-name=*string*	Assigns the value of *string* to the alias *alias-name*.

If no operands are given, all alias definitions are written to standard output.

The format for displaying aliases is as follows:

"<*name*>=<*value*>\n"

The *value* string is written with appropriate quoting so that it is suitable for reinput to the shell.

NOTES

The alias utility is a built-in utility to the standard shell (sh(1)). An executable version of this utility is available in /usr/bin/alias.

The csh(1) utility has a built-in alias utility that has slightly different characteristics. See csh(1).

EXIT STATUS

The alias utility exits with one of the following values:

0 Successful completion.

>0 One of the *alias-name* operands specified did not have an alias definition or an error occurred.

SEE ALSO

csh(1), sh(1), unalias(1)

NAME

amlaw – Displays maximum theoretical parallel processing speedups

SYNOPSIS

amlaw [*ncpu* [*pc-values*]]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The amlaw utility displays the maximum theoretical speedup for selected pairs (number of CPUs, percent parallelism) using Amdahl's Law.

The amlaw utility accepts the following arguments:

ncpu *ncpu* is the number of CPUs; it must be greater than 1.

pc-values *pc-values* is the percent parallelism; each value must be greater than 0 and less than or equal to 99.9999999. (*pc-values* that are less than 1 are multiplied by 100.0.) Up to 19 *pc-values* may be used.

A table of sample speedups is generated, using selected *ncpu* and *pc-values*, when any of the following conditions occur:

- No arguments are specified.
- Only *ncpu* is specified.
- *ncpu* is 0.
- The only *pc-value* specified is 0.
- More than one *pc-value* is specified.

EXAMPLES

The following command line generates the speedup information shown below:

```
$ amlaw  8  94.45
8 CPUs, 94.45% Parallelism:  Max Theoretical speedup is  5.76
```

The following command line generates the speedup table shown below:

```
$ amlaw  16  97  97.1  97.2  97.3  97.4  97.5
        Max Theoretical Speedup for 16 CPUs

          %   Speedup   |      %   Speedup
        ----  -------   |     ---  -------
        97.0   11.034   |    97.3  11.388
        97.1   11.150   |    97.4  11.511
        97.2   11.268   |    97.5  11.636
```

SEE ALSO

Optimizing Code on Cray PVP Systems, Cray Research publication SG–2192

NAME

apropos – Locates commands by keyword

SYNOPSIS

apropos *keyword* ...

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The apropos utility displays the man page name, section number, and a short description for each man page that contains the *keyword* in its NAME line. This information is contained in the /usr/man/whatis file. Each keyword is considered separately and the case of letters is ignored. Words that contain *keyword* also are displayed; for example, if *keyword* is compile, apropos displays all instances of compiler also.

The apropos utility accepts the following operand:

keyword The character string that is searched for in the NAME line of available man pages.

The function of the apropos utility is identical to that of the -k option of the man(1) utility.

If the line displayed by apropos starts *filename*(*section*)..., you can type man *section filename* to display the man page for *filename*.

FILES

/usr/man/whatis Database

SEE ALSO

man(1), whatis(1)

NAME

ar – Archive and library maintainer for portable archives

SYNOPSIS

```
ar -d [-l] [-s] [-v] archive files
ar -m [-a] [-b] [-i] [-l] [-s] [-v] [posname] archive files
ar -p [-s] [-v] archive [files]
ar -q [-c] [-l] [-s] [-u] [-v] archive files
ar -r [-a] [-b] [-c] [-i] [-l] [-s] [-u] [-v] [posname] archive files
ar -t [-s] [-v] [-z] archive [files]
ar -x [-o] [-s] [-v] [-z] [-C] [-T] archive [files]
```

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4
AT&T extensions (-a, -b, -i, -l, -m, -o, -q, and -s options)

DESCRIPTION

The ar utility maintains groups of files combined into a single archive file. Once an archive has been created, new files can be added, and existing files can be extracted, deleted, or replaced. The string and the file headers used by ar consist of printable ASCII characters. If an archive is composed of printable files, the entire archive is printable.

When ar creates an archive, it creates headers in a format that is portable across all Cray Research systems. The portable archive format and structure are described in detail in the ar(5) man page.

The following options are supported:

-a Places new modules after *posname*. This option only works when the -r or -m options are specified.

-b Places new modules before *posname*. This option only works when the -r or -m options are specified.

-c Suppresses the diagnostic message that is written to standard error by default when the archive file *archive* is created.

-d Deletes *files* from *archive*.

-i Equivalent to -b.

-l Places temporary files in the local (current working) directory rather than in the default temporary directory, TMPDIR.

`-m` Moves the named *files* to the end of the archive. If `-a`, `-b`, or `-i` are specified, the *posname* argument must be present and, as in `-r`, specify where the files are to be moved.

`-o` Sets the "last modified" date to the date recorded in the archive. This option only works if the `-x` option also is specified.

`-p` Writes the contents of the files from *archive* to the standard output. If no *files* are specified, the contents of all files in the archive are written in the order of the archive.

`-q` Quickly appends the named *files* to the end of the archive file. The positioning options `-a`, `-b`, and `-i` are not valid. No checks are performed as to whether or not the added members are already in the archive. This option is useful to avoid quadratic behavior when creating a large archive piece-by-piece.

`-r` Replaces or adds *files* to *archive*. If the archive named by *archive* does not exist, a new archive file is created. Files that replace existing files do not change the order of the archive.

If the `-a`, `-b`, or `-i` options are not specified, *files* that do not replace existing files are appended to the archive. Otherwise, the *posname* operand must be present and *files* are positioned according to the option and *posname* specified.

`-s` Forces regeneration of the archive symbol table, even if `ar` is not invoked with a command that will modify the archive contents (see `ranlib`(1)).

`-t` Writes a table of contents of *archive* to the standard output. The files specified by the *file* operands are included in the written list. If no *file* operands are specified, all files in *archive* are included in the order of the archive.

`-u` Updates older files. When used with the `-r` or `-m` options, files within the archive will be replaced only if the corresponding *file* has a modification time that is at least as new as the modification time of the file within the archive.

`-v` Gives verbose output. When used with the option characters `-d`, `-m`, `-r`, or `-x`, writes a detailed file-by-file description of the archive creation and maintenance activity.

When used with `-p`, writes the name of the file to the standard output before writing the file itself to the standard output.

When used with `-t`, includes a long listing of information about the files within the archive.

`-x` Extracts the files named by the *files* operands from *archive*. The contents of the archive file are not changed. If no *files* operands are given, all files in the archive are extracted. If the file name of a file extracted from the archive is longer than that supported in the directory to which it is being extracted, the results are undefined. The modification time of each file extracted is set to the time the file is extracted from the archive.

`-z` Provides backward-compatibility for those archives incorrectly constructed in UNICOS 7.0 and UNICOS 8.0. This option is used when the message `malformed archive` is generated. If this message is produced using the `-z` option, invoke `ar` again without the option. If this message still appears, the archive is truly malformed. This option can only be used with the `-t` and `-x` options.

`-C` Prevents extracted files from replacing existing files of the same name.

`-T` Allows file name truncation of extracted files whose archive names are longer than the file system can support. By default, extracting a file with a name that is too long is an error; a diagnostic message is written and the file is not extracted.

The following operands are supported:

posname An archive member name used as a reference point in positioning other files in the archive.

archive The path name of the archive file.

files Path names. Only the last component is used when comparing against the names of files in the archive. If two or more *files* operands have the same last path name component (base name), both are added to the archive. In the case of such files, however, each file operand matches only the first archive file having a name that is the same as the last component of the file operand. The archive format used by `ar` does not truncate valid file names of files added to, or replaced in, the archive.

NOTES

All *files* operands can be path names. However, files within archives are identified by a file name, which is the last component of the path name used when the file was entered into the archive. The comparison of *files* operands to the names of files in the archives is performed by comparing the last component of the operand to the name of the archive file.

ENVIRONMENT VARIABLES

Variable	Description
`RANLIB`	Absolute path name of the utility that optimizes the archive file. The default is `opt/ctl/bin/ranlib`. If set to a `NULL` string, optimization of the archive is not performed.
`RANLIBFLAGS`	Option arguments to the executable file specified by `RANLIB`.

EXIT STATUS

The `ar` utility exits with one of the following values:

0 Successful completion.

>0 An error occurred.

MESSAGES

A phase error indicates that one or more files of the archive are no longer available, and that the archive itself can no longer be used. It is possible that some individual parts of the archive may be salvaged by extracting them.

Most other error messages are self-explanatory.

EXAMPLES

Example 1: The following example creates library `mylib.a` from object files `file1.o`, `file2.o`, and `file3.o`. After the creation of the library, the library's table of contents is printed along with verbose output:

```
$ ar -r mylib.a file1.o file2.o file3.o
$ ar -tv mylib.a
```

Example 2: The following example adds `newfile.o` after `file1.o` in the archive `mylib.a` with verbose output:

```
$ ar -rav file1.o mylib.a newfile.o
```

Example 3: The following example extracts all files from the archive `mylib.a`:

```
$ ar -x mylib.a
```

FILES

`TMPDIR`	Directory containing temporary files for user

SEE ALSO

`bld`(1), `segldr`(1),

`stat`(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

`tmpnam`(3C) in the *UNICOS System Libraries Reference Manual*, Cray Research publication SR–2080

`a.out`(5), `ar`(5), `relo`(5) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

`ranlib`(1) Online only

NAME

asa – Interprets ASA carriage control characters

SYNOPSIS

asa [*files*]

IMPLEMENTATION

All Cray Research systems
SPARC systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The asa utility interprets the output of any programs that use ASA carriage control characters. It processes either the *files* that are given as arguments or the standard input if you do not supply any file names. The first character of each line is assumed to be a control character; the control characters have the following meanings:

<space>	The rest of the line is output without change.
0	A <newline> is output, followed by the rest of the input line. This results in double spacing.
-	Two <newline> symbols are output, followed by the rest of the input line. This results in triple spacing.
1	Advance to the top of the next page, then output the rest of the input line.
+	Return to the column position 1; then overwrite the previous line with the rest of the input line.

Lines beginning with other than the preceding characters are treated as if they began with a <space>. The first character of a line is **not** printed. If any such lines appear, an appropriate diagnostic will appear on standard error. This program forces the first line of each input file to start on a new page.

NOTES

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
system, secadm	Allowed to interpret any input file. In a privileged administrator shell environment, shell-redirected I/O is not subject to file protections.
sysadm	Allowed to interpret any input file subject to security label restrictions. Shell-redirected I/O is subject to security label restrictions.

If the PRIV_SU configuration option is enabled, the super user is allowed to interpret any input file. Shell-redirected I/O on behalf of the super user is not subject to file protections.

EXIT STATUS

The asa utility exits with one of the following values:

0 All input files were output successfully.

>0 An error occurred.

EXAMPLES

Example 1: The following example uses asa as a filter to view the output of a Fortran program that uses ASA carriage control characters. myfort is the Fortran executable file, datafile is the input data to myfort, and textfile is the output file:

```
$ myfort < datafile | asa > textfile
```

Example 2: asa can be used with an executable file as follows:

```
$ a.out | asa > file1
```

The output, properly formatted and paginated, would be directed to a file file1. Fortran output sent to the file could be viewed by entering the following:

```
$ asa file
```

SEE ALSO

fsplit(1), lp(1), nasa(1)

NAME

ascheck – Validates the array services configuration

SYNOPSIS

ascheck [-F] [-Kl *key*] [-Kr *key*] [-p *port*] [-q ...] [-s *server* [-D]] [-t *value*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The ascheck command validates the configuration of all the array services daemons known to the local array services daemon. ascheck checks to ensure that the various array and machine definitions are consistent, both on the individual servers and with corresponding definitions on the other servers. If any problems are found, they are described in detail, and suggestions are provided to correct the situation.

It should be noted that ascheck is not a syntax checker for the individual array services configuration files; that function is handled by the -c option of arrayd(8). Instead, ascheck is used to check the semantics of a syntactically correct array services configuration file.

The ascheck command relies on information provided by array services. Thus, for complete coverage, an array services daemon should be running on every machine that is mentioned in the local configuration file(s).

The checks that ascheck makes include the following:

- Ensuring that the local array services daemon is accessible.
- Verifying that all of the array services daemons known to the local daemon are also available.
- Ensuring that each array services daemon is using the correct version of the array services library.
- Ensuring that the LOCAL IDENT value used by a particular array services daemon is not different from the machine ID used by the operating system on that daemon's machine.
- Suggesting that a machine ID be set in the operating system on those systems that have an array services daemon but have not yet set a machine ID.
- Ensuring that no two arrays on a particular server have the same ARRAY IDENT value.
- Ensuring that the SERVER IDENT value, which may be declared for a particular ARRAY/MACHINE definition, matches the LOCAL IDENT value of the corresponding server.
- Warning if the list of machines defined for a particular array does not match the list of machines for an array with the same name on a different server.
- Warning about any server that has two or more arrays defined but has not specified a default array.

The ascheck command takes several options, primarily to control the selection of the local server and to select the desired amount of output.

-F	When used with -s, indicates that array services requests should be forwarded to the specified server via the server on the current machine rather than sent directly.
-Kl *key*	Use *key* for the local authentication key when communicating directly with a remote array services daemon.
-Kr *key*	Use *key* for the remote authentication key when communicating directly with a remote array services daemon.
-p *port*	Specifies the port address of the local arrayd server. Defaults to the value of the ARRAYD_PORT environment variable if present, or 5434 otherwise.
-q	Produces less verbose (quieter) output. Repeated occurrences (either -q -q . . . or -qq . . .) may further decrease the amount of output.
-s *server*	Specifies the host name or IP address of the local arrayd server. Defaults to the value of the ARRAYD environment variable if present, or localhost otherwise.
-D	When used with -s, indicates that array services requests should be sent directly to the specified server, rather than being forwarded to that server by the array services daemon running on the current machine. This is the default behavior.
-t *value*	Specifies the time-out value (in seconds) used for waiting on individual responses from the array daemon.

NOTES

The array services daemon (arrayd(8)) must be running on all machines that are to be examined. It does not necessarily have to be running on the machine that executes ascheck if an alternate server was specified in some way.

SEE ALSO

arrayd(8)

arrayd.conf(5) Online only

NAME

at, batch – Executes commands at a later time

SYNOPSIS

```
at [-m] [-f file] [-q queuename] -t time
at [-m] [-f file] [-q queuename] timespec ...
at -r at_job_id ...
at -l -q queuename
at -l [at_job_id ...]
batch
```

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The at and batch utilities read commands from standard input to be executed at a later time. The at utility lets you specify when the commands should be executed; jobs queued with batch execute when system load level permits.

Standard output and standard error output are mailed to the user unless they are redirected elsewhere. Exported shell environment variables (except TMPDIR), the current directory, umask, and ulimit are retained when the commands are executed. Open file descriptors, traps, and priority are lost.

Users whose names appear in the /usr/lib/cron/at.allow file are permitted to use at. When that file does not exist, the /usr/lib/cron/at.deny file is checked to determine whether a user should be denied access to at. If neither file exists, only root will be allowed to submit a job. Null file at.allow means that no users are allowed to use at; null file at.deny means that no users are denied the use of at. The allow/deny files consist of one user name per line.

The at utility accepts the following options:

-f *file* Specifies the path name of a file to be used as the source of the at job, rather than standard input.

-l Lists all jobs scheduled for the invoking user if no *at_job_id* operands are specified. If *at_job_id*s are specified, lists only information for these jobs.

-m Sends mail to the invoking user after the *at-job* has run, announcing its completion. Standard output and standard error produced by the *at-job* are mailed to the user as well, unless redirected elsewhere. Mail is sent even if the job produces no output. If you omit -m, the job's standard output and standard error are mailed to the user, unless redirected elsewhere.

`-q` *queuename*

Queues the command in queue *queuename*. The default queue is `a`. *queuename* can be one of 25 different queues (that is, `a`, `b`, `d–z`.). Queue `b` is defined as the batch queue; jobs in this queue run whenever the system administrator-defined maximum level is not exceeded. Jobs in all other queues run at the time specified on the command line. The character `c` is not allowed after the `-q` option. When used with the `-l` option, the search is limited to that particular queue.

`-r` Removes the jobs with the specified *at_job_id* operands that were previously scheduled by the `at` utility.

`-t` *time* Submits the job to be run at the specified *time*. The option-argument is of the form:

[[*CC*]*YY*]*MMDDhhmm*[*.SS*]

where each two digits represent the following:

CC First two digits of the year (the century).

YY Second two digits of the year.

MM Month of the year (01–12).

DD Day of the month (01–31).

hh Hour of the day (00–23).

mm Minute of the hour (00–59).

SS Second of the minute (00–61).

If you do not specify *CC* or *YY*, the current year is assumed. If you specify *YY* but not *CC*, *CC* will become 19, if *YY* is in the range 69–99. *CC* will become 20, if *YY* is in the range 00–68.

timespec Consists of a *time* followed by an optional *date* and an optional *increment*, or the special name, `now`.

You may specify the time as 1, 2, or 4 digits. A 1-digit or 2-digit number indicates hours. A 4-digit number indicates hours and minutes. Alternatively, you may alternatively specify time as two numbers separated by a colon, meaning *hour*`:`*minute*. An `am` or `pm` suffix may be appended; otherwise, 24-hour clock time is understood. The suffix `gmt`, `utc` or `zulu`, may be used to indicate GMT. The special names `noon` and `midnight` are also recognized.

An optional date may be specified as either a month name followed by a day number (and possibly year number preceded by an optional comma) or a day of the week (fully spelled or abbreviated to 3 characters). Two special "days," `today` and `tomorrow`, are recognized. If date is not specified, and if the given hour is greater than the current hour, `today` is assumed. `tomorrow` is assumed if the specified hour is less than the current hour. If the specified month is less than the current month (and no year is given), next year will be assumed.

The optional increment is simply a number suffixed by one of the following: `minutes`, `hours`, `days`, `weeks`, `months`, or `years`. (The singular form is also accepted.) The keyword `next` is equivalent to an increment number of `+1`. For example, the following are equivalent commands:

```
at 2pm +1 week
at 2pm next week
```

Valid time specifiers include the following:

```
0815am Jan 24
8:15am Jan 24
now +1 day
5 pm Friday
```

`at` and `batch` write the job number and schedule time to standard error.

`batch` submits a batch job. It is equivalent to `at -q b -m now`.

`at -r` removes jobs previously scheduled by `at` or `batch`. The job number is the number given to you previously by the `at` or `batch` utilities. You can also get job numbers by typing `at -l`. Unless you are the super user, you can remove only your own jobs.

At the time of submission, `at` jobs are run at the user's current security label.

NOTES

If this utility is installed with a privilege assignment list (PAL), a user who is assigned the following privilege text upon execution of this command is allowed to perform the action shown:

Privilege Text	Action
`showall`	Allowed to manage all jobs.

If this utility is installed with a PAL, a user with one of the following active categories is allowed to perform the action shown:

Active Category	Action
`system`, `secadm`, `sysadm`, `sysops`	Allowed to manage all jobs.

If the `PRIV_SU` configuration option is enabled, the super user is allowed to manage all jobs.

EXIT STATUS

The `at` utility exits with one of the following values:

0 The `at` utility successfully submitted, removed, or listed a job or jobs.

>0 An error occurred.

The batch utility exits with one of the following values:

0 The batch utility successfully submitted a job.

>0 An error occurred.

MESSAGES

The at utility reports various syntax errors and times out-of-range.

EXAMPLES

Example 1: The at and batch utilities read from standard input the commands to be executed at a later time. sh(1) provides various ways of specifying standard input. Within your commands, it may be useful to redirect standard output.

The following sequence can be used at a terminal:

```
$ batch
make filename >outfile
<CONTROL-d>
```

Example 2: The following sequence, which demonstrates the redirecting of standard error to a pipe, is useful in a shell procedure (the sequence of output redirection specifications is significant):

```
$ batch <<!
make filename 2>&1 >outfile | mail loginid
!
```

Example 3: To use the at utility to schedule the execution of a shell script or executable binary file (stored in *execfile*), use shell input redirection (<), as in the following example:

```
$ at 0815am Jan 24 < execfile
```

Example 4: To have a job reschedule itself, invoke at from within the shell procedure by including in the shell file code similar to the following:

```
$ echo "sh shellfile" | at 1900 thursday next week
```

FILES

/usr/lib/cron	Main cron directory
/usr/lib/cron/at.allow	List of allowed users
/usr/lib/cron/at.deny	List of denied users
/usr/lib/cron/queue	Scheduling information
/usr/spool/cron/atjobs	Spool area

SEE ALSO

`kill`(1), `mail`(1), `nice`(1), `ps`(1), `sh`(1)

`chown`(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

`proto`(5), `queuedefs`(5) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

`cron`(8) in the *UNICOS Administrator Commands Reference Manual*, Cray Research publication SR–2022

NAME

awk, nawk – Pattern scanning and processing language

SYNOPSIS

awk [-F *ERE*] [-v *assignment*]... *program* [*argument*]...

awk [-F *ERE*] -f *progfile*... [-v *assignment*]... [*argument*]...

nawk [-F *ERE*] [-v *assignment*]... *program* [*argument*]...

nawk [-F *ERE*] -f *progfile*... [-v *assignment*]... [*argument*]...

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The awk utility scans each input file for lines that match any of a set of patterns specified in *program*. Each pattern in *program* may have an associated action that will be performed when a line of a file matches the pattern. The set of patterns may appear either literally as *program* or in a file specified by using the -f option. This version of awk provides capabilities that were not available with previous versions.

awk accepts the following options and arguments:

-F *ERE* Defines the input field separator to be the Extended Regular Expression *ERE*. (Currently, only Basic Regular Expressions are supported.)

-v *assignment*... *assignment*s in the form *x=xvalue y=yvalue* can be passed to awk; *x* and *y* are awk built-in variables. The specified variable assignment occurs prior to executing the awk program, including the actions associated with BEGIN patterns, if any. You can specify multiple occurrences of the -v option.

-f *progfile* File that contains the set of pattern-action statements. If you specify multiple instances of this option, the concatenation of the files specified as *progfile* is used as the awk program.

program Set of patterns for which awk scans file. To protect the *program* string from the shell, enclose it in a single quotation marks.

argument Either of the following types of *argument*s can be specified:

assignment *assignment*s in the form *x=xvalue y=yvalue* can be passed to awk; *x* and *y* are awk built-in variables.

file Input files. If no files exist, the standard input is read. The file name `-` specifies the standard input. Each input line is matched against the pattern portion of every pattern-action statement; the associated action is performed for each matched pattern.

An input line is usually made up of fields separated by white space. (To changed the default, use the `FS` built-in variable or the `-F` *ERE* option.) The fields are denoted `$1`, `$2`, ...; `$0` refers to the entire line.

A pattern-action statement has the form:

pattern `{` *action* `}`

You can omit either pattern or action. If no action with a pattern exists, the matching line is printed. If no pattern with an action exists, the action is performed on every input line.

Patterns are arbitrary Boolean combinations (`!`, `||`, `&&`, and parentheses) of relational expressions and Extended Regular Expressions. A relational expression is one of the following:

"*expression relop expression*"
"*expression matchop regular_expression*"
expression `in` *array-name*
`(`*expression*`,`*expression*`, ... )` `in` *array-name*

A *relop* is any of the six relational operators in C, and a *matchop* is either `~` (contains) or `!~` (does not contain). An *expression* is an arithmetic expression, a relational expression, the special expression

var `in` array

or a Boolean combination of these.

You can use the special patterns `BEGIN` and `END` to capture control before the first input line has been read and after the last input line has been read, respectively. These keywords do not combine with any other patterns.

Regular expressions are as in `egrep` (see `egrep`(1)). In patterns, they must be surrounded by slashes. Isolated Extended Regular Expressions in a pattern apply to the entire line. Regular expressions may also occur in relational expressions. A pattern may consist of two patterns separated by a comma; in this case, the action is performed for all lines between an occurrence of the first pattern and the next occurrence of the second pattern.

You can use an Extended Regular Expression to separate fields by using the `-F` *ERE* option or by assigning the expression to the built-in variable `FS`. The default is to ignore leading `<blank>`s and to separate fields by `<blank>`s and/or `<tab>` characters. However, if `FS` is assigned a value, leading `<blank>`s are no longer ignored.

Other built-in variables include the following:

`ARGC`	Command-line argument count
`ARGV`	Command-line argument array
`FILENAME`	Name of the current input file

FNR	Ordinal number of the current record in the current file
FS	Input field separator Extended Regular Expression (default is <blank> and <tab> characters)
NF	Number of fields in the current record
NR	Ordinal number of the current record
OFMT	Output format for numbers (default is %.6g)
OFS	Output field separator (default is <blank> character)
ORS	Output record separator (default is <newline> character)
RS	Input record separator (default is <newline>)
SUBSEP	Separates multiple subscripts (default is 034)

An action is a sequence of statements. A statement can be one of the following:

```
if ( expression ) statement [ else statement ]
while ( expression ) statement
do statement while ( expression )
for ( expression ; expression ; expression ) statement
for ( var in array ) statement
delete array [subscript]
break
continue
{ [ statement ] ... }
expression          # commonly variable = expression
print [ expression-list ] [ >expression ]
printf format [ , expression-list ] [ >expression ]
next                # skip remaining patterns on this input line
exit [expr] # skip the rest of the input; exit status is expr
return [expr]
```

Statements are terminated by semicolons, <newline> characters, or right braces. An empty expression-list stands for the whole input line. Expressions take on string or numeric values as appropriate, and are built using the operators +, -, *, /, % and concatenation (indicated by a <blank>). The operators ++, --, +=, -=, *=, /=, and %= are also available in expressions. Variables may be scalars, array elements (denoted *x*[*i*]), or fields. Variables are initialized to the null string or 0. Array subscripts can be any string, not necessarily numeric, allowing for associative memory. String constants are quoted (").

The print statement prints its arguments on the standard output, or on a file if >*expression* is present, or on a pipe if | *cmd* is present. The current output field separator separates arguments. The output record separator terminates arguments. The printf statement formats its expression list according to the format in printf(3C).

awk has a variety of built-in functions: arithmetic, string, input/output, and general.

The arithmetic functions are atan2, cos, exp, int, log, rand, sin, sqrt, and srand. int truncates its argument to an integer. rand returns a random number between 0 and 1. srand(expr) sets the seed value for rand to expr or uses the time of day if expr is omitted.

The string functions are as follows:

gsub(*ERE*,*repl*,[*in*])
: Behaves like the sub string function, except that it replaces successive occurrences of the Extended Regular Expression (such as the ed(1) global substitute command) in $0 or in the *in* argument, when specified.

index(*s*,*t*)
: Returns the position in string *s* where string *t* first occurs, or 0 if it does not occur at all.

int
: Truncates to an integer value.

length[([*s*])]
: Returns the length of its argument taken as a string, or of the whole line, $0, if no argument exists.

match(*s*,*ERE*)
: Returns the position in string *s* where the Extended Regular Expression occurs, or 0 if it does not occur at all. RSTART is set to the starting position (which is the same as the returned value), and RLENGTH is set to the length of the matched string.

rand
: Random number on (0, 1).

split(*s*,*a*,*fs*)
: Splits the string *s* into array elements *a*[1], *a*[2], . . ., *a*[*n*], and returns *n*. The separation is done with the Extended Regular Expression *fs* or with the field separator FS if *fs* is not given.

srand
: Sets the seed for rand

sprintf(*fmt*,*expr*,*expr*,...)
: Formats the expressions according to the printf(3C) format given by *fmt* and returns the resulting string.

sub(*ERE*,*repl*,*in*)
: Substitutes the string *repl* in place of the first instance of the Extended Regular Expression *ERE* in string *in* and returns the number of substitutions. If you omit *in*, awk substitutes in the current record ($0).

substr(*s*,*m*,*n*)
: Returns the *n* -character substring of *s* that begins at position *m*.

tolower(*s*)
: Return a string based on the string *s*. Each character in *s* that is an uppercase letter specified to have a tolower mapping by the LC_CTYPE category of the current locale is replaced in the returned string by the lowercase letter specified by its mapping. Other characters in *s* are unchanged in the returned string.

toupper(*s*)
: Return a string based on the string *s*. Each character in *s* that is a lowercase letter is specified to have a toupper mapping by the LC_CTYPE category of the current locale is replaced in the returned string by the uppercase letter specified by its mapping. Other characters in *s* are unchanged in the returned string.

The input/output and general functions are as follows:

`close(`*filename*`)`	Closes the file or pipe named *filename*.
cmd `| getline`	Pipes the output of *cmd* into `getline`; each successive call to *getline* returns the next line of output from *cmd*.
`getline`	Sets `$0` to the next input record from the current input file.
`getline <`*file*	Sets `$0` to the next record from *file*.
`getline` *x*	Sets variable *x* instead.
`getline` *x* `<`*file*	Sets *x* from the next record of *file*.
`system(`*cmd*`)`	Executes *cmd* and returns its exit status.

All forms of `getline` return 1 for successful input, 0 for end of file, and −1 for an error.

`awk` also provides user-defined functions. You can define such functions (in the pattern position of a pattern-action statement) as follows:

```
function name(args,...) { stmts }
func name(args,...) { stmts }
```

Function arguments are passed by value if scalar and by reference if array name. Argument names are local to the function; all other variable names are global. You can nest function calls, and functions may be recursive. You can use the `return` statement to return a value.

NOTES

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
`system`, `secadm`	In a privileged administrator shell environment, shell-redirected I/O is not subject to file protections.
`sysadm`	Shell-redirected output is subject to security label restrictions.

If the `PRIV_SU` configuration option is enabled, shell-redirected I/O on behalf of the super user is not subject to file protections.

WARNINGS

Currently, only Basic Regular Expressions are supported. If fields are involved, input white space is not preserved on output.

There are no explicit conversions between numbers and strings. To force an expression to be treated as a number, add 0 to it; to force it to be treated as a string, concatenate the null string (`""`) to it.

Separate pattern-action statements by either a semicolon or a `<newline>`. This is an incompatibility with the old version of `awk`.

EXIT STATUS

The awk utility exits with one of the following values:

0 All input files were processed successfully.

>0 An error occurred.

The exit status can be altered within the program by using an exit expression.

EXAMPLES

Example 1: The following awk program prints lines longer than 72 characters:

```
length > 72
```

Example 2: The following awk program prints first two fields in opposite order:

```
{ print $2, $1 }
```

Example 3: The following awk program prints the first two fields in opposite order with input fields separated by comma and/or <blank>s and <tab>s:

```
BEGIN { FS = ",[ \t]*|[ \t]+" }
      { print $2, $1 }
```

Example 4: The following awk program adds up the first column, and then prints the sum and average:

```
      { s += $1 }
END   { print "sum is", s, " average is", s/NR }
```

Example 5: The following awk program prints fields in reverse order:

```
{ for (i = NF; i > 0; --i) print $i }
```

Example 6: The following awk program prints all lines between start/stop pairs:

```
/start/, /stop/
```

Example 7: The following awk program prints all lines whose first field differs from the previous one:

```
$1 != prev { print; prev = $1 }
```

Example 8: The following awk program simulates the echo(1) utility:

```
BEGIN {
      for (i = 1; i < ARGC; i++)
            printf "%s", ARGV[i]
      printf "\n"
      exit
      }
```

Example 9: The following awk program prints a file, filling in page numbers starting at 5:

```
/Page/      { $2 = n++; }
      { print }
```

Assuming this program is in a file named prog, the following command line prints the file input and numbers its pages starting at 5:

```
awk -f prog n=5 input
```

SEE ALSO

echo(1), ed(1), egrep(1), grep(1), lex(1), sed(1)

printf(3C) in the *UNICOS System Libraries Reference Manual*, Cray Research publication SR–2080

The AWK Programming Language, Aho, Kerninghan, Weinberger, Addison-Wesley, 1988

sed & awk, Dale Doherty, O'Reilly & Associates, Inc., 1990

NAME

banner – Makes posters

SYNOPSIS

banner *strings*

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The banner utility prints the *strings* argument (each string can be up to 10 characters long) in large letters on standard output. You can include spaces in an argument by surrounding them with quotation marks. The maximum number of characters that can be accommodated in a line is implementation-dependent; excess characters are simply ignored.

EXAMPLES

Example 1: Outputs the words `hi there` as one string:

```
banner "hi there"
```

Example 2: Outputs `hi` as one string and `there` as a second string.

```
banner hi there
```

SEE ALSO

echo(1)

NAME

basename, dirname – Prints parts of path names on standard output

SYNOPSIS

basename *string* [*suffix*]

dirname *string*

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The basename utility deletes any prefix ending in / and the *suffix* (if present in *string*) from *string*, and prints the result on the standard output.

The dirname utility converts *string* to the name of the directory that contains the file name corresponding to the last pathname component in *string*. The result is printed on the standard output.

NOTES

The *suffix* must match identically to the suffix in *string*.

EXIT STATUS

The basename and dirname utilities exit with one of the following values:

0 Successful completion.

>0 An error occurred.

EXAMPLES

Example 1: The following shell script, invoked with the /usr/src/cmd/cat.c argument, compiles the specified file and moves the output to a file named cat in the current directory:

```
cc $1
mv a.out $(basename "$1" .c)
```

Example 2: The following example sets shell variable NAME to /usr/src/cmd:

```
NAME=$(dirname /usr/src/cmd/cat.c)
```

SEE ALSO

sh(1)

NAME

bc – An arbitrary precision calculator language

SYNOPSIS

bc [-l] [-w] [-s] [*file* ...]

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4
FSF XPG4 (-s and -w options)

DESCRIPTION

bc is a language that supports arbitrary precision numbers with interactive execution of statements. This bc utility replaces the bc utility that was included in UNICOS releases prior to UNICOS 8.3. The old bc utility has been renamed obc(1).

There are some similarities in the bc syntax to the C programming language. A standard math library is available by command-line option. If requested, the math library is defined before processing any files. bc starts by processing code from all the files listed on the command line in the order listed. After all files have been processed, bc reads from the standard input. All code is executed as it is read. (If a file contains a command to halt the processor, bc will never read from the standard input.)

This version of bc contains several extensions beyond traditional bc implementations and the POSIX draft standard. Command-line options can cause these extensions to print a warning or to be rejected. This document describes the language accepted by this processor. Extensions will be identified as such.

The bc utility accepts the following options and operand:

-l Defines the standard math library.

-w Gives warnings for extensions to POSIX bc.

-s Processes exactly the POSIX bc language.

file The file or files to be processed.

The most basic element in bc is the number. Numbers are arbitrary precision numbers. This precision is both in the integer part and the fractional part. All numbers are represented internally in decimal and all computation is done in decimal. (This version truncates results from divide and multiply operations.) There are two attributes of numbers, the *length* and the *scale*. The length is the total number of significant decimal digits in a number and the scale is the total number of decimal digits after the decimal point. For example:

```
.000001 has a length of 6 and scale of 6.
1935.000 has a length of 7 and a scale of 3.
```

Numbers are stored in two types of variables, simple variables and arrays. Both simple variables and array variables are named. Names begin with a letter followed by any number of letters, digits and underscores. All letters must be lowercase. (Full alphanumeric names are an extension. In POSIX `bc`, all names are a single lowercase letter.) The type of variable is clear by the context, because all array variable names will be followed by brackets (`[]`).

There are four special variables, *scale*, *ibase*, *obase*, and *last*. *scale* defines how some operations use digits after the decimal point. The default value of *scale* is 0. *ibase* and *obase* define the conversion base for input and output numbers. The default for both input and output is base 10. *last* (an extension) is a variable that has the value of the last printed number. These will be discussed in further detail where appropriate. All of these variables may have values assigned to them as well as used in expressions.

Comments in `bc` start with the characters `/*` and end with the characters `*/`. Comments may start anywhere and appear as a single space in the input. (This causes comments to delimit other input items. For example, a comment can not be found in the middle of a variable name.) Comments include any newlines (end-of-line) between the start and the end of the comment.

Expressions and Statements

The numbers are manipulated by expressions and statements. Since the language was designed to be interactive, statements and expressions are executed as soon as possible. There is no "main" program. Instead, code is executed as it is encountered. (Functions, discussed in detail later, are defined when encountered.)

A simple expression is just a constant. `bc` converts constants into internal decimal numbers using the current input base, specified by the variable *ibase*. (There is an exception in functions.) The legal values of *ibase* are 2 through 16 (F). Assigning a value outside this range to *ibase* will result in a value of 2 or 16. Input numbers may contain the characters 0–9 and A–F. (NOTE: They must be uppercase. Lowercase letters are variable names.) Single-digit numbers always have the value of the digit, regardless of the value of `ibase` (for example, A = 10). For multidigit numbers, `bc` changes all input digits greater or equal to *ibase* to the value of *ibase*–1. This makes the number `FFF` always be the largest 3-digit number of the input base.

Full expressions are similar to many other high-level languages. Since there is only one kind of number, there are no rules for mixing types. Instead, there are rules on the scale of expressions. Every expression has a scale. This is derived from the scale of original numbers, the operation performed and in many cases, the value of the variable *scale*. Legal values of the variable *scale* are 0 to the maximum number representable by a C integer.

In the following descriptions of legal expressions, *expr* refers to a complete expression and *var* refers to a simple or an array variable. A simple variable is just a *name* and an array variable is specified as *name*[*expr*]. Unless specifically mentioned, the *scale* of the result is the maximum scale of the expressions involved.

`-` *expr* The result is the negation of the expression.

`++` *var* The variable is incremented by one and the new value is the result of the expression.

`--` *var* The variable is decremented by one and the new value is the result of the expression.

var ++ The result of the expression is the value of the variable, and then the variable is incremented by one.

var -- The result of the expression is the value of the variable, and then the variable is decremented by one.

expr + *expr* The result of the expression is the sum of the two expressions.

expr - *expr* The result of the expression is the difference of the two expressions.

expr * *expr* The result of the expression is the product of the two expressions.

expr / *expr* The result of the expression is the quotient of the two expressions. The scale of the result is the value of the variable *scale*.

expr % *expr* The result of the expression is the "remainder," and it is computed in the following way. To compute *a*%*b*, first *a*/*b* is computed to *scale* digits. That result is used to compute *a*-(*a*/*b*)**b* to the scale of the maximum of *scale*+*scale*(b) and *scale*(a). If *scale* is set to zero, and both expressions are integers, this expression is the integer remainder function.

expr ^ *expr* The result of the expression is the value of the first raised to the second. The second expression must be an integer. (If the second expression is not an integer, a warning is generated, and the expression is truncated to get an integer value.) The scale of the result is *scale* if the exponent is negative. If the exponent is positive, the scale of the result is the minimum of the scale of the first expression times the value of the exponent and the maximum of *scale* and the scale of the first expression (for example, *scale*(*a*^*b*) = *min*(*scale*(*a*)***b*, *max*(*scale*, *scale*(*a*))). It should be noted that *expr*^0 will always return the value of 1.

(*expr*) This alters the standard precedence to force the evaluation of the expression.

var = *expr* The variable is assigned the value of the expression.

var <op>= *expr* This is equivalent to *var* = *var* <op> *expr* with the exception that the *var* part is evaluated only once. This can make a difference if *var* is an array.

Relational expressions are a special kind of expression that always evaluate to 0 or 1, 0 if the relation is false and 1 if the relation is true. These may appear in any legal expression. (POSIX bc requires that relational expressions are used only in if, while, and for statements and that only one relational test may be done in them.) The relational operators are as follows:

expr1 < *expr2* The result is 1 if *expr1* is strictly less than *expr2*.

expr1 <= *expr2* The result is 1 if *expr1* is less than or equal to *expr2*.

expr1 > *expr2* The result is 1 if *expr1* is strictly greater than *expr2*.

expr1 >= *expr2* The result is 1 if *expr1* is greater than or equal to *expr2*.

expr1 == *expr2* The result is 1 if *expr1* is equal to *expr2*.

expr1 != *expr2* The result is 1 if *expr1* is not equal to *expr2*.

Boolean operations are also legal. (POSIX bc does **not** have boolean operations). The result of all boolean operations are 0 and 1 (for false and true) as in relational expressions. The boolean operators are as follows:

!*expr* The result is 1 if *expr* is 0.

expr && *expr* The result is 1 if both expressions are nonzero.

expr || *expr* The result is 1 if either expression is nonzero.

The expression precedence is as follows: (lowest to highest)

```
|| operator, left associative
&& operator, left associative
! operator, nonassociative
Relational operators, left associative
Assignment operator, right associative
+ and - operators, left associative
*, / and % operators, left associative
^ operator, right associative
unary - operator, nonassociative
++ and -- operators, nonassociative
```

This precedence was chosen so that POSIX-compliant bc programs will run correctly. This will cause the use of the relational and logical operators to have some unusual behavior when used with assignment expressions. Consider the expression:

```
a = 3 < 5
```

Most C programmers would assume this would assign the result of 3 < 5 (the value 1) to the variable *a*. What this does in bc is assign the value 3 to the variable *a* and then compare 3 to 5. It is best to use parentheses when using relational and logical operators with the assignment operators.

A few more special expressions are provided in bc. These have to do with user-defined functions and standard functions. They all appear as *name*(*parameters*). See the Functions subsection for user-defined functions. The standard functions are as follows:

length (*expression*)
: The value of the length function is the number of significant digits in the expression.

read ( )
: The read function (an extension) will read a number from the standard input, regardless of where the function occurs. WARNING: This can cause problems with the mixing of data and program in the standard input. The best use for this function is in a previously written program that needs input from the user, but never allows program code to be input from the user. The value of the read function is the number read from the standard input using the current value of the variable ibase for the conversion base.

`scale ( expression )`
: The value of the `scale` function is the number of digits after the decimal point in the expression.

`sqrt ( expression )`
: The value of the `sqrt` function is the square root of the expression. If the expression is negative, a run-time error is generated.

Statements

Statements (as in most algebraic languages) provide the sequencing of expression evaluation. In `bc`, statements are executed "as soon as possible." Execution happens when a newline is encountered, and there is one or more complete statement. Due to this immediate execution, newlines are very important in `bc`. In fact, both a semicolon and a newline are used as statement separators. An improperly placed newline will cause a syntax error. Because newlines are statement separators, it is possible to hide a newline by using the backslash character (\). The sequence \<*nl*>, where <*nl*> is the newline, appears to `bc` as white space instead of a newline. A statement list is a series of statements separated by semicolons and newlines. The following is a list of `bc` statements and what they do. (Items enclosed in brackets (`[]`) are optional parts of the statement.)

`expression`
: This statement does one of two things. If the expression starts with <*variable*> <*assignment*> ..., it is considered to be an assignment statement. If the expression is not an assignment statement, the expression is evaluated and printed to the output. After the number is printed, a newline is printed. For example, *a*`=1` is an assignment statement, and `(`*a*`=1)` is an expression that has an embedded assignment. All numbers that are printed are printed in the base specified by the variable *obase*. The legal values for *obase* are 2 through `BC_BASE_MAX`. (See the Limits subsection.) For bases 2 through 16, the usual method of writing numbers is used. For bases greater than 16, `bc` uses a multicharacter digit method of printing the numbers, in which each higher base digit is printed as a base-10 number. The multicharacter digits are separated by spaces. Each digit contains the number of characters required to represent the base-10 value of *obase*`-1`. Since numbers are of arbitrary precision, some numbers may not be printable on a single output line. These long numbers will be split across lines using the \ as the last character on a line. The maximum number of characters printed per line is 70. Due to the interactive nature of `bc`, printing a number causes the side effect of assigning the printed value to the special variable *last*. This allows the user to recover the last value printed without having to retype the expression that printed the number. Assigning to *last* is legal and will overwrite the last printed value with the assigned value. The newly assigned value will remain until the next number is printed or another value is assigned to *last*.

`string`
: The string is printed to the output. Strings start with double quotation marks and contain all characters until the next double quotation marks. All characters are taken literally, including any newline. No newline character is printed after the string.

`print` *list* The `print` statement (an extension) provides another method of output. The *list* is a list of strings and expressions separated by commas. Each string or expression is printed in the order of the list. No terminating newline is printed. Expressions are evaluated and their value is printed and assigned the variable `last`. Strings in the `print` statement are printed to the output and may contain special characters. Special characters start with the backslash character (\). The special characters recognized by `bc` are `b` (bell), `f` (form feed), `n` (newline), `r` (carriage return), `t` (tab), and \ (backslash). Any other character following the backslash will be ignored. This still does not allow the double quote character to be part of any string.

`{ statement_list }`
This is the compound statement. It allows multiple statements to be grouped together for execution.

`if (`*expression*`) then` *statement1* `[else` *statement2*`]`
The `if` statement evaluates the expression and executes *statement1* or *statement2* depending on the value of the expression. If the expression is nonzero, *statement1* is executed. If *statement2* is present, and the value of the expression is 0, then *statement2* is executed. (The `else` clause is an extension.)

`while (`*expression*`)` *statement*
The `while` statement will execute the statement while the expression is nonzero. It evaluates the expression before each execution of the statement. Termination of the loop is caused by a zero expression value or the execution of a break statement.

`for ([`*expression1*`] ; [`*expression2*`] ; [`*expression3*`])` *statement*
The `for` statement controls repeated execution of the statement. *expression1* is evaluated before the loop. *expression2* is evaluated before each execution of the statement. If it is nonzero, the statement is evaluated. If it is zero, the loop is terminated. After each execution of the statement, *expression3* is evaluated before the reevaluation of *expression2*. If *expression1* or *expression3* is missing, nothing is evaluated at the point at which it would be evaluated. If *expression2* is missing, it is the same as substituting the value 1 for *expression2*. (The optional expressions are an extension. POSIX `bc` requires all three expressions.) The following is equivalent code for the `for` statement:

```
expression1;
while (expression2) {
   statement;
   expression3;
}
```

`break` The `break` statement causes a forced exit of the most recent enclosing `while` statement or `for` statement.

`continue` The `continue` statement (an extension) causes the most recent enclosing for *statement* to start the next iteration.

`halt` The `halt` statement (an extension) is an executed statement that causes the `bc` processor to quit only when it is executed. For example, `if (0 == 1) halt` will not cause `bc` to terminate, because the halt is not executed.

`return` Returns the value 0 from a function. (See the Functions subsection.)

`return (`*expression*`)`
Returns the value of the expression from a function. (See the Functions subsection.)

Pseudo Statements

These statements are not statements in the traditional sense. They are not executed statements. Their function is performed at "compile" time.

`limits` Prints the local limits enforced by the local version of `bc`. This is an extension.

`quit` When the `quit` statement is read, the `bc` processor is terminated, regardless of where the `quit` statement is found. For example, `if (0 == 1)`, `quit` will cause `bc` to terminate.

`warranty` Prints a longer warranty notice. This is an extension.

Functions

Functions provide a method of defining a computation that can be executed later. Functions in `bc` always compute a value and return it to the caller. Function definitions are *dynamic* in the sense that a function is undefined until a definition is encountered in the input. That definition is then used until another definition function for the same name is encountered. The new definition then replaces the older definition. A function is defined as follows:

`define` *name* `(`*parameters*`) {` *newline*
auto_list *statement_list* `}`

A function call is just an expression of the form *name*`(`*parameters*`)`.

Parameters are numbers or arrays (an extension). In the function definition, zero or more parameters are defined by listing their names separated by commas. Numbers are only call-by-value parameters. Arrays are only call-by-variable. Arrays are specified in the parameter definition by the notation *name*`[]`. In the function call, actual parameters are full expressions for number parameters. The same notation is used for passing arrays as for defining array parameters. The named array is passed by variable to the function. Since function definitions are dynamic, parameter numbers and types are checked when a function is called. Any mismatch in number or types of parameters will cause a run-time error. A run-time error will also occur for the call to an undefined function.

The *auto_list* is an optional list of variables that are for local use. The syntax of the auto list (if present) is `auto` *name*`, ... ;`. (The semicolon is optional.) Each *name* is the name of an auto variable. Arrays may be specified by using the same notation as used in parameters. These variables have their values pushed onto a stack at the start of the function. The variables are then initialized to zero and used throughout the execution of the function. At function exit, these variables are popped so that the original values (at the time of the function call) of these variables are restored. The parameters are really auto variables that are initialized to a value provided in the function call. Auto variables are different than traditional local variables in that if function A calls function B, B may access function A's auto variables by just using the same name, unless function B has called them auto variables. Due to the fact that auto variables and parameters are pushed onto a stack, `bc` supports recursive functions.

The function body is a list of `bc` statements. Again, statements are separated by semicolons or newlines. Return statements cause the termination of a function and the return of a value. There are two versions of the return statement. The first form, `return`, returns the value 0 to the calling expression. The second form, `return (`*expression*`)`, computes the value of the expression and returns that value to the calling expression. There is an implied `return (0)` at the end of every function. This allows a function to terminate and return 0 without an explicit return statement.

Functions also change the usage of the variable *ibase*. All constants in the function body will be converted using the value of *ibase* at the time of the function call. Changes of *ibase* will be ignored during the execution of the function except for the standard function `read`, which will always use the current value of *ibase* for conversion of numbers.

Math Library

If `bc` is invoked with the `-l` option, a math library is preloaded and the default scale is set to 20. The math functions will calculate their results to the scale set at the time of their call. The math library defines the following functions:

`s (`*x*`)` The sine of *x* in radians.

`c (`*x*`)` The cosine of *x* in radians.

`a (`*x*`)` The arctangent of *x*.

`l (`*x*`)` The natural logarithm of *x*.

`e (`*x*`)` The exponential function of raising `e` to the value *x*.

`j (`*n*`,`*x*`)` The Bessel function of integer order *n* of *x*.

Differences

This version of `bc` was implemented from the POSIX P1003.2/D11 draft and contains several differences and extensions relative to the draft and traditional implementations. It is not implemented in the traditional way using `odc`(1). This version is a single process which parses and runs a byte code translation of the program. There is an undocumented option (`-c`) that causes the program to output the byte code to the standard output instead of running it. It was mainly used for debugging the parser and preparing the math library.

A major source of differences is extensions, where a feature is extended to add more functionality, and additions, where new features are added. The following is the list of differences and extensions.

`LANG` environment
: This version does not conform to the POSIX standard in the processing of the `LANG` environment variable and all environment variables starting with `LC_`.

`names`
: Traditional and POSIX `bc` have single–letter names for functions, variables and arrays. They have been extended to be multicharacter names that start with a letter and may contain letters, numbers, and the underscore character.

`Strings`
: Strings are not allowed to contain NUL characters. POSIX says all characters must be included in strings.

`last`
: POSIX `bc` does not have a *last* variable.

`comparisons`
: POSIX `bc` allows comparisons only in the `if` statement, the `while` statement, and the second expression of the `for` statement. Also, only one relational operation is allowed in each of those statements.

`if statement, else clause`
: POSIX `bc` does not have an `else` clause.

`for statement`
: POSIX `bc` requires all expressions to be present in the `for` statement.

`&&, ||, !`
: POSIX `bc` does not have the logical operators.

`read function`
: POSIX `bc` does not have a read function.

`print statement"`
: POSIX `bc` does not have a `print` statement.

`continue statement`
: POSIX `bc` does not have a `continue` statement.

`array parameters`
: POSIX `bc` does not have `array` parameters. Other implementations of `bc` may have call by value array parameters.

`=+, =-, =*, =/, =%, =^`
: POSIX `bc` does not require these "old style" assignment operators to be defined. This version may allow these "old style" assignments. Use the `limits` statement to see if the installed version supports them. If it does support the "old style" assignment operators, the statement `a =- 1` will decrement `a` by 1 instead of setting `a` to the value –1.

`spaces in numbers`
: Other implementations of `bc` allow spaces in numbers. For example, `x=1 3` would assign the value 13 to the variable *x*. The same statement would cause a syntax error in this version of `bc`.

errors and execution

This implementation varies from other implementations in terms of what code will be executed when syntax and other errors are found in the program. If a syntax error is found in a function definition, error recovery tries to find the beginning of a statement and continue to parse the function. Once a syntax error is found in the function, the function will not be callable and becomes undefined. Syntax errors in the interactive execution code will invalidate the current execution block. The execution block is terminated by an end-of-line that appears after a complete sequence of statements. For example,

```
a = 1
b = 2
```

has two execution blocks and

```
{ a = 1
  b = 2 }
```

has one execution block. Any run-time error will terminate the execution of the current execution block. A run-time warning will not terminate the current execution block.

Interrupts — During an interactive session, the SIGINT signal (usually generated by the <CONTROL-c> character from the terminal) will cause execution of the current execution block to be interrupted. It will display a run-time error indicating which function was interrupted. After all run-time structures have been cleaned up, a message will be printed to notify the user that bc is ready for more input. All previously defined functions remain defined, and the value of all nonauto variables are the value at the point of interruption. All auto variables and function parameters are removed during the cleanup process. During a noninteractive session, the SIGINT signal will terminate the entire run of bc.

Limits

The following are the limits currently in place for this bc processor. Some of them may have been changed by an installation. Use the limits statement to see the actual values.

BC_BASE_MAX — The maximum output base is currently set at 999. The maximum input base is 16.

BC_DIM_MAX — This is currently an arbitrary limit of 65,535 as distributed. Your installation may be different.

BC_SCALE_MAX — The number of digits after the decimal point is limited to INT_MAX digits. Also, the number of digits before the decimal point is limited to INT_MAX digits.

BC_STRING_MAX — The limit on the number of characters in a string is INT_MAX characters.

exponent — The value of the exponent in the raise operation (^) is limited to LONG_MAX.

multiply — The multiply routine may yield incorrect results if a number has more than LONG_MAX / 90 total digits. For 32-bit longs, this number is 23,860,929 digits.

`code size`	Each function and the "main" program are limited to 10,240 bytes of compiled byte code each. This limit (`BC_MAX_SEGS`) can be changed easily to have more than 10 segments of 1024 bytes.
`variable names`	The current limit on the number of unique names is 32,767 for each of simple variables, arrays, and functions.

EXAMPLES

Example 1: In `/bin/sh`, the following will assign the value of pi to the shell variable `pi`.

```
pi=$(echo "scale=10; 4*a(1)" | bc -l)
```

Example 2: The following is the definition of the exponential function used in the math library. This function is written in POSIX bc.

```
scale = 20

/* Uses the fact that e^x = (e^(x/2))^2
   When x is small enough, we use the series:
     e^x = 1 + x + x^2/2! + x^3/3! + . . .
*/

define e(x) {
  auto  a, d, e, f, i, m, v, z

  /* Check the sign of x. */
  if (x<0) {
    m = 1
    x = -x
  }

  /* Precondition x. */
  z = scale;
  scale = 4 + z + .44*x;
  while (x > 1) {
    f += 1;
    x /= 2;
  }

  /* Initialize the variables. */
  v = 1+x
  a = x
  d = 1

  for (i=2; 1; i++) {
    e = (a *= x) / (d *= i)
    if (e == 0) {
      if (f>0) while (f--)  v = v*v;
      scale = z
      if (m) return (1/v);
      return (v/1);
    }
    v += e
  }
}
```

Example 3: The following is code that uses the extended features of bc to implement a simple program for calculating checkbook balances. This program is best kept in a file so that it can be used many times without having to retype it at every use.

```
scale=2
print "\nCheck book program!\n"
print "  Remember, deposits are negative transactions.\n"
print "  Exit by a 0 transaction.\n\n"

print "Initial balance? "; bal = read()
bal /= 1
print "\n"
while (1) {
  "current balance = "; bal
  "transaction? "; trans = read()
  if (trans == 0) break;
  bal -= trans
  bal /= 1
}
quit
```

The following is the definition of the recursive factorial function:

```
define f (x) {
  if (x <= 1) return (1);
  return (f(x-1) * x);
}
```

FILES

In most installations, bc is completely self-contained. Where executable size is of importance or the C compiler does not deal with very long strings, bc will read the standard math library from the file /usr/lib/lib.b.

DIAGNOSTICS

If any file on the command line cannot be opened, bc will report that the file is unavailable and terminate. Also, there are compile-time and run-time diagnostics that should be self-explanatory.

SEE ALSO

awk(1), obc(1), odc(1)

NAME

bdiff – Compares very large files for differences

SYNOPSIS

bdiff *file1 file2* [*n*] [-s]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The bdiff utility is used in a manner analogous to that of diff(1) to find lines that must be changed in two files to bring them into agreement. Its purpose is to allow the processing of files that are too large for diff. The bdiff utility ignores lines common to the beginning of both files, splits the remainder of each file into *n*-line segments, and invokes diff upon corresponding segments.

The bdiff utility accepts the following arguments and options:

file1 First file to compare; required.

file2 Second file to compare; required. If *file1* or *file2* is -, bdiff reads the standard input.

n Causes bdiff to split remainder of file into *n*-line segments, rather than the default 3500-line segments. *n* must be numeric. This option is useful when 3500-line segments are too large for diff, causing bdiff to fail.

-s Specifies that diagnostics are not to be printed by bdiff (however, this does not suppress possible exclamations by diff(1)).

If both optional arguments are specified, they must appear in the order indicated in the SYNOPSIS section.

The bdiff utility's output is exactly like that of diff, with line numbers adjusted to account for the segmenting of the files (that is, to make it look as if the files had been processed whole). Because of the segmenting of the files, bdiff does not necessarily find the smallest sufficient set of file differences.

NOTES

The bdiff utility will exit with a nonzero status even if there are differences between the input files. This behavior is necessary because bdiff is invoked by delta(1).

MESSAGES

Use help(1) for explanations.

FILES

`/tmp/bd`*name* Temporary working file

SEE ALSO

`delta`(1), `diff`(1), `help`(1)

NAME

bftp – Provides user interface to the background file transfer program

SYNOPSIS

/usr/ucb/bftp

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The bftp command is the user interface to the background file transfer program (BFTP). You can use bftp to submit a request for a future file transfer by using the standard Internet file transfer protocol (FTP), which is described in RFC 959.

Because bftp makes use of third-party FTP, the source and the destination hosts do not have to be operational at the time the request is submitted. Transfers are scheduled locally through the system batch processor using the at(1) utility. Therefore, the exact time at which a file transfer occurs depends on how often jobs are run from the system batch queue.

For more information on BFTP, see RFC 1068.

The bftp command recognizes the following commands:

append true | false
: Appends the transferred file to the existing file if the destination file exists and this option is true. If this option is false and the file name is already in use, the existing file is replaced if write permission on the destination host is enabled.

command copy | delete | move
: Sets the source file disposition to copy, move (which is equivalent to copy and delete), or simply delete the source file. The default is copy. These commands set the file transfer mode.

d-dir *destination-directory*
: Establishes the destination directory into which the file will be transferred. The directory name must end with a directory-delimiter character.

d-file *filename*
: Sets the destination file name, which is the name of the file when the transfer is complete.

d-host *hostname*
: Sets the destination host name, which is the host to which the file will be transferred.

d-password *password*
: Specifies the password on the destination host.

`d-user` *user*
: Specifies the user name (login) on the destination host.

`explain` Displays a short explanation on the use of `bftp`.

`find` Locates and displays a previously submitted BFTP job. `bftp` prompts for the (optional) *RequestID* and the *RequestKeyword*. After a request is located and displayed, you can change and resubmit it, or cancel it.

`help-all`
: Displays a description of each command.

`help` [*command*]
: Prints an informative message about the meaning of *command*. If you do not specify an argument, `bftp` prints a list of the commands and their descriptions.

`init` Discards all information about a request. This command resets `bftp` to the default startup values.

`interval`
: Displays starting retry interval (in minutes) and number of retries.

`mailbox` *mailbox-name*
: Sends the completion notification message to this address.

`mode stream | block | compress`
: Sets the FTP mode. The default value is `stream`.

`multiple true | false`
: When `true`, transfers all files that match the pattern set in `s-file` to be transferred. Matching is performed on the source host.

`prompt` Prompts for all of the necessary information to set up a transfer.

`quit` Ends the current `bftp` session.

`r-delete` *request-file*
`r-list`
`r-load` *request-file*
`r-store` *request-file*
: The request commands manage request files, which are used to save BFTP requests for future use.

`s-acct` and `d-acct`
: Sets the login account name for the transaction. Usually, this option is not needed.

`s-dir` *source-directory*
: Sets the source directory from which the file will be transferred. The directory name must end with a directory-delimiter character.

`s-file` *source-filename*
: Sets the source file name of the file to be transferred.

`s-host` *hostname*
Sets the source host name, which is the host from which the file will be transferred.

`s-password` *password*
Specifies the password on the source host.

`s-port | d-port` *n*
Sets the FTP transaction source or destination port number, which is usually 21.

`s-user` *user*
Specifies the user name (login) on the source host.

`show` Displays the current parameter values.

`simple-example`
Displays an example that shows how to submit a request.

`stru file | record | page`
Sets the FTP structure.

`submit` Places the request in the batch queue. `bftp` prompts for the *StartTime* and the *RequestKeyword*.

`time` *StartTime*
Sets the date and time at which the file will be transferred.

`transfer`
Performs the requested transfer now.

`type image | ascii | ebcdic | local | binary`
Sets the FTP file type. The `ascii` and `ebcdic` types have further parameters of `nonprint`, `telnet`, and `carriage-control`. The default type is `ascii nonprint`. The representation type may be one of network ASCII, EBCDIC, image, or local byte size with a specified byte size (usually for PDP-10s and PDP-20s). The network ASCII and EBCDIC types have a further subtype that specifies whether vertical format control (new-line characters, form feeds, and so on) will be passed through (nonprint), provided in TELNET format, or provided in ASA carriage control format.

`unique true | false`
Sets unique mode. If `unique` is `true`, the destination file name is guaranteed to be unique. This prevents the replacement of old files with new files of the same name.

`verbose true | false`
Sets verbose mode. When `verbose` is `true`, `bftp` prints each command used in the transaction during the verify and the transfer operations.

`verify` Validates the request. The current request is checked to determine whether or not all needed information was entered. Then `bftp` connects to the specified hosts to determine whether or not the specified parameters are supported.

FILES

The bftp command creates the following files that keep track of requests that are in progress:

b*123456789*.cmd
b*123456789*.lis
b*123456789*.msg
b*123456789*.req

The following files are saved by using the r-store command:

s.*request-name*

Usually, bftp stores request files in the home directory of the user who is logged on. To have bftp store these files in another directory, use the system setenv command to set $BFTPDIR. For example:

```
setenv BFTPDIR ~yourname/.bftp
```

SEE ALSO

at(1), crontab(1), ftp(1B)

cron(8), ftpd(8) in the *UNICOS Administrator Commands Reference Manual*, Cray Research publication SR–2022

NAME

bg – Runs jobs in the background

SYNOPSIS

bg [*job_id* ...]

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The bg utility resumes suspended jobs from the current shell execution environment (see sh(1)) by running the jobs as background jobs. If the job specified by *job_id* is already a running background job, the bg utility has no effect and exits immediately.

Using bg to place a job into the background causes its process ID to become "known in the current shell execution environment," as if it had been started as an asynchronous list.

The bg utility supports the following operand:

job_id Specifies the job to be resumed as a background job. If you omit *job_id*, the most recently suspended job is used. For a description of the *job_id* format, see sh(1).

For information about running a job in the foreground, see fg(1).

NOTES

The bg utility is a built-in utility to the standard shell (sh(1)). An executable version of this utility is available in /usr/bin/bg.

EXIT STATUS

The bg utility exits with one of the following values:

0 Successful completion.

>0 An error occurred.

SEE ALSO

fg(1), jobs(1), sh(1)

NAME

bld – Maintains relocatable libraries

SYNOPSIS

bld *key*[*opts*] [*position-obj*] *bldname* [*args*]
bld R *bldname old-obj-name new-obj-name*

IMPLEMENTATION

Cray PVP systems

DESCRIPTION

The bld utility collects relocatable modules into a relocatable library. The library can be maintained and manipulated with the bld utility.

The bld utility line has two formats as shown in the SYNOPSIS section. The first format is used to physically manipulate modules in a library. The second fomrat is used to change the names of modules in the library; only the <R> key (without *opts*) is used in this version.

A *position-obj* or any other object is defined as *file*:*module*. *file* may be a full path name. For example, the following refers to the module MODULE that came from the file file.o:

```
file.o:MODULE
```

In the example, file.o (or file.o:) refers to all modules in the file file.o, and :MODULE refers to the module MODULE, which has no file name associated with it. If MODULE contains a '$', the '$' must be escaped so it is protected from being expanded by the shell.

The bld utility accepts the following arguments:

key Keyletters are as follows:

- d Deletes the named objects from the library.
- m Moves the named modules to the end of the library. If a positioning character is present, the *position-obj* argument must be present and specifies where the modules are to be moved.
- p Prints the named modules in the library in binary format to stdout.
- q Takes the relocatable modules specified in *args* and appends them to the end of the library. Optional positioning characters are not valid. The command does not check whether the added modules are already in the library.
- r Takes the relocatable modules specified in *args* and replaces them in the library.

  WARNING: The command does not check for duplicate names in the argument list.

`t` Prints a table of contents of the library. If no names are specified, all modules in the library are used for the table of contents. If names are specified, only those modules are used for the table of contents.

`x` Extracts the named modules. If no names are specified, all modules in the library are extracted. The modules are placed in a file whose name is the same as the *file* or, if none, the *module*. In neither case does `x` alter the library file.

`R` Renames an object. (This key is used only with the second form of the command line; no *opts* may be used.) This lets users change the name of an object or part of the name of an object. For example:

`bld R bld.a foo.o bar.o` changes the file name of every module that came from `foo.o` to `bar.o`.

`bld R bld.a foo.o:BAR bar.o` changes the file name associated with the module `BAR` to `bar.o`.

`bld R bld.a :BAR foo.o:BAR` adds the file name `foo.o` to module `BAR`.

`bld R bld.a foo.o:BAR :BAR` removes the file name `foo.o` from module `BAR`

`S` Keeps files in sync and ensures that there is a one-to-one mapping with an `ar`(1) library. It may add, replace, or delete modules in the build library based on the `ar` library.

opts Option letters are as follows:

`a` Places new modules after *position-obj*. This option works only with the `<r>` and `<m>` keys.

`b` Places new modules before *position-obj*. This option works only with the `<r>` and `<m>` keys.

`c` Suppresses the message that is produced by default when *bldname* is created.

`e` Displays all entry points in the specified modules. This option works only with the `<t>` key.

`i` Places new modules before *position-obj*. This option works only with the `<r>` and `<m>` keys.

`l` Places temporary files in the local current working directory `/tmp`. By default, temporary files are placed in the directory specified by `TMPDIR`.

`v` Gives a verbose module-by-module description of the making of a new library file from the old library and the constituent modules. When used with `t`, this option gives a long listing of all information about the modules. When used with `x`, it precedes each module with a name.

`z` Inhibits the association of a file from which the module comes. This feature allows for the replacement of a module of the same name from a different compilation unit (`.o`). This is useful in Fortran applications in which subroutines are generated by `cft77` as multiple module units. This option may be used only when the `r` and `q` options are specified. This option should be used only to emulate the operation of the COS `BUILD` utility. Its use is inappropriate within the context of a UNICOS makefile or nmakefile. See the EXAMPLES section.

position-obj Specifies that new modules are to be placed after (`a`) or before (`b` or `i`) *position-obj*. Otherwise, new files are placed at the end. *position-obj* is required when the `a`, `b`, or `i` options are used.

bldname Specifies the library. If the library file *bldname* exists, it must be a `bld`-formatted archive. If the `q`, `r`, or `s` keys are specified and *bldname* does not exist, `bld` will create the library *bldname*.

args Names of files containing relocatable modules when using the `r` or `q` keys. Otherwise, names of modules in the library, *bldname*.

old-obj-name Old name of the module.

new-obj-name New name of the module.

NOTES

The `bld` utility has the following limitations:

- Currently, the `bld` utility does not interpret CRAY T3D object files. Use `ar`(1) to process CRAY T3D object files.
- If the file names in the `bld` archive have subdirectories and if the subdirectories do not exist, `bld` will error exit when extracting files from a `bld` archive.
- When the default (no `z` *opt*) is used, a file name is *permanently* associated with the one or more modules contained in the file. Once generated into a `bld` archive, the file name is retained, even after extraction. To clear the file name from the module, use the `<R>` key.
- The `-z` option is not supported with Fortran 90.
- If `bld` cannot access one of the files in an argument list, then none of the files will be added to the library. This behavior differs from the `ar`(1) utility, which adds all of the files on the argument list that can be successfully accessed.

EXAMPLES

Example 1: The following example shows the use of the `z` *opt* in a Fortran application in which subroutines are generated by as multiple module units. In this example, `filea.o` contains the modules `SUBA` and `SUBB`:

```
$ bld qvz bldname filea.o
```

Example 2: By compiling file `fileb.f`, which contains subroutine modules `SUBB` and `SUBC`, `SUBB` and `SUBC` can be replaced using the following command:

```
$ bld rvz bldname fileb.o
```

This will overlay `SUBB` even though it was created in a different `.o` file. When the default (no `z` *opt*) is used, a file name is *permanently* associated with the one or more modules contained in the file. Once generated into a `bld` archive, the file name is retained, even after extraction. To clear the file name from the module, use the `<R>` key.

Example 3: The following example shows a variety of `bld` uses. The shell prompt is indicated by a `$`.

```
$ ls
malloc.c memmgr.c memmgr.h
$ cc -c malloc.c
$     #Initial insertion of malloc.o
$ bld q lib.a malloc.o
bld: creating lib.a
$ cc -c memmgr.c
$     #Initial insertion of memmgr.o
$ bld q lib.a memmgr.o
$     #Table of contents
$ bld t lib.a
malloc.o:malloc$c
memmgr.o:memmgr$c
$     #Verbose table of contents
$ bld tv lib.a
  Date      Time    Compiler  Version   OS Vers.  Object Name
01/27/91  10:59:28  Std C     2X065406    7.0     malloc.o:malloc$c
01/27/91  11:00:41  Std C     2X065406    7.0     memmgr.o:memmgr$c
$     #Assume that malloc.c has changes
$ cc -c malloc.c
$     #Verbose replace
$ bld rv lib.a malloc.o
r - malloc.o:malloc$c
$     #Verbose table of contents
$ bld tv lib.a
  Date      Time    Compiler  Version   OS Vers.  Object Name
01/27/91  11:04:08  Std C     2X065406    7.0     malloc.o:malloc$c
01/27/91  11:00:41  Std C     2X065406    7.0     memmgr.o:memmgr$c
$     #Notice the updated time of only malloc.o
$ rm malloc.o memmgr.o
$ ls
lib.a malloc.c memmgr.c memmgr.h
$     #Extraction of memmgr.o
$ bld x lib.a memmgr.o
```

```
$ ls
lib.a malloc.c memmgr.c memmgr.h memmgr.o
$     #The extraction did not modify lib.a
$ bld tv lib.a
  Date      Time    Compiler  Version   OS Vers.  Object Name
01/27/91  11:04:08  Std C     2X065406    7.0     malloc.o:malloc$c
01/27/91  11:00:41  Std C     2X065406    7.0     memmgr.o:memmgr$c
$     #Move malloc.o after memmgr.o
$ bld ma memmgr.o:memmgr\$c lib.a malloc.o
$ bld tv lib.a
  Date      Time    Compiler  Version   OS Vers.  Object Name
01/27/91  11:00:41  Std C     2X065406    7.0     memmgr.o:memmgr$c
01/27/91  11:04:08  Std C     2X065406    7.0     malloc.o:malloc$c
$     #Removing memmgr.o
$ bld d lib.a memmgr.o
$ bld tv lib.a
  Date      Time    Compiler  Version   OS Vers.  Object Name
01/27/91  11:04:08  Std C     2X065406    7.0     malloc.o:malloc$c
```

FILES

TMPDIR/bld*	Temporary files

BUGS

If the same file is mentioned twice in an argument list, it may be put in the library twice.

SEE ALSO

ar(1), nm(1), segldr(1)

a.out(5), bld(5), relo(5) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

NAME

builddefs – Reads a definitions file that has embedded keywords to produce a keyword file and a definitions file without embedded keywords

SYNOPSIS

builddefs *infile deffile*

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The builddefs utility reads a definitions file that has embedded keywords to produce a keyword file and a definitions file without embedded keywords. A pair of keywords in the form ++*term* marks each definition in the input file. Synonyms for the term, if any, also begin with ++ and follow the keyword line (do not mark synonyms in pairs). You should enter all keywords and synonyms into the definitions file and use the correct capitalization conventions. The keywords and synonyms are recorded in the keyword file in lowercase characters.

Example:

```
++access control list
++ACL
  An access control list (ACL) contains the individual/group
  identifiers and the individual/group access modes for the subjects
  who will be allowed to access the file.  See also discretionary
  access control.
++access control list
```

The builddefs utility checks all keywords for keywording errors. The following rules apply to keywording:

- The two ++ symbols that appear in columns 1 and 2 of the intermediate definitions file identify a keyword. The keyword immediately follows the ++ symbols, with no intervening blank spaces or tabs. Empty keywords (that is, ++ with no following text) are not allowed.
- A keyword consists of up to 48 characters. If a keyword is longer than 48 characters, it will be truncated.
- Each definition must have two keywords (a matching pair). The first keyword indicates the start of the definition. The second keyword indicates the end of the definition.
- Synonyms for a keyword are in the form ++*synonym* and are limited to 48 characters. Do not mark synonyms in pairs.

The builddefs utility accepts the following arguments:

infile Input file.

deffile Output file.

Running the builddefs utility produces the output file *deffile* and a keyword file *deffile*_k.

NOTES

The builddefs utility lets you modify the Cray Research definitions file (CRAYdefs_i) or add local definitions files. The following subsections describe these procedures.

Modifying the Cray Research Definitions File

The following procedure shows how to use builddefs to modify the Cray Research definitions file that the define(1) utility uses.

If you modify the Cray Research definitions file, your changes will be lost during the installation of a UNICOS revision or update. In this case, back up the modified definitions file, install the UNICOS revision or update, and then reapply the modified definitions file.

1. Copy the CRAYdefs_i definitions file, which has embedded keywords, from the default definitions directory, /usr/lib/define, to your working directory.
2. Edit the file to make the desired changes.
3. Run builddefs on the edited file as follows:

   ```
   builddefs CRAYdefs_i CRAYdefs
   ```

   CRAYdefs_i is the input file, and CRAYdefs is the output file. This utility produces two files: CRAYdefs, which is a definitions file without embedded keywords, and CRAYdefs_k, which is a keyword file.
4. Set the DEFINEDIR environment variable to the working directory that contains the CRAYdefs and CRAYdefs_k files. For the Korn and standard shells, set the variable as follows:

   ```
   DEFINEDIR=directoryname
   export DEFINEDIR
   ```

   For the C shell, set the variable as follows:

   ```
   setenv DEFINEDIR directoryname
   ```
5. Test the modified file by using the define(1) utility on selected terms.
6. Install the CRAYdefs_i, CRAYdefs, and CRAYdefs_k files in the /usr/lib/define directory.
7. If necessary, reset the DEFINEDIR environment variable. For the Korn and standard shells, reset the variable as follows:

   ```
   unset DEFINEDIR
   ```

For the C shell, reset the variable as follows:

```
unsetenv DEFINEDIR
```

Creating a Local Definitions File

The following procedure shows how to use `builddefs` to create a local definitions file for use by the `define`(1) utility. When multiple definitions files are in the definitions directory, the `define`(1) utility reads the files in alphabetical order; that is, it will search file `aaaa` before file `bbbb`.

Your local definitions files, installed in the default `define` directory `/usr/lib/define`, might be removed during the installation of a UNICOS revision or update if your site begins the installation process from a clean partition. In this case, back up your local definitions files, install the UNICOS revision or update, and then reinstall your local definitions files. Prepare an input file that contains embedded keywords according to the keywording rules contained in the DESCRIPTION section.

1. Run `builddefs` on the local file as follows:

   `builddefs` *sitedefs*`_i` *sitedefs*

   The *sitedefs*`_i` argument is the input file, and *sitedefs* is the output file. This command produces two files: *sitedefs*, which is a definitions file without embedded keywords, and *sitedefs*`_k`, which is a keyword file.

3. If you want to test how your local file works, set the `DEFINEDIR` environment variable to the working directory (*directoryname*) that contains the new formatted definition file. If you do not want to test the local file, skip to step 6. For the Korn and standard shells, set the variable as follows:

   ```
   DEFINEDIR=directoryname
   export DEFINEDIR
   ```

   For the C shell, set the variable as follows:

   `setenv DEFINEDIR` *directoryname*

4. Copy the `CRAYdefs` file by using the following command:

   `cp CRAYdefs CRAYdefs_k` *directoryname/*

5. Test the new file by using the `define`(1) utility on selected terms.
6. Install the *sitedefs* and *sitedefs*`_k` files in the `/usr/lib/define` directory. The `define`(1) utility reads the files in this directory and searches them in sequence for search string matches.
7. If necessary, reset the `DEFINEDIR` environment variable. For the Korn and standard shells, reset the variable as follows:

   ```
   unset DEFINEDIRR
   ```

   For the C shell, reset the variable as follows:

   ```
   unsetenv DEFINEDIR
   ```

FILES

`builddefs.cat`	`builddefs` message catalog
`builddefs.exp`	`builddefs` explain message catalog
`builddefs.msg`	`builddefs` message text file

SEE ALSO

`define`(1)

General UNICOS System Administration, Cray Research publication SG–2301

NAME

`cal` – Prints calendar

SYNOPSIS

`cal` [[*month*] *year*]

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The `cal` utility prints a calendar for the specified year. If a month is also specified, a calendar just for that month is printed. If neither is specified, a calendar for the present month is printed. *year* can be from 1 through 9999. *month* is a number from 1 through 12. The calendar produced is for England and the United States.

NOTES

The year is always considered to start in January.

Beware that `cal` 83 refers to the early Christian era, not the 20th century.

EXIT STATUS

The following exit values are returned:

0 Successful completion.

>0 An error occurred.

EXAMPLES

An unusual calendar is printed for September 1752. That is the month 11 days were skipped to make up for lack of leap year adjustments. To see this calendar, type the following:

```
cal 9 1752
```

NAME

calendar – Reminder service

SYNOPSIS

```
calendar
```

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The `calendar` utility consults the `calendar` file in your directory and prints out lines that contain today's or tomorrow's date anywhere in the line. Most reasonable month-day dates such as `Aug. 24`, `august 24`, or `8/24` are recognized, but not `24 August` or `24/8`. On weekends, ''tomorrow'' extends through Monday.

When you specify the argument, `calendar` does its job for all users who have a file `calendar` in their login directory and sends them any positive results by `mail`(1). Usually, this is done daily by administrative facilities in the UNICOS operating system.

EXIT STATUS

The following exit values are returned:

0 Successful completion.

>0 An error occurred.

BUGS

Your calendar must have public permission (see `chmod`(1)) for you to get reminder service.

The extended idea of ''tomorrow'' does not account for holidays.

EXAMPLES

In this example, suppose you have previously created a file named `calendar` that contains the following lines:

```
2/2    Report A due
2/3    Department meeting
2/4    Time card
2/4    Lunch with Lori
2/5    Report B due
```

If you enter the `calendar` command on February 3, you will receive the following output:

```
2/3    Department meeting
2/4    Time card
2/4    Lunch with Lori
```

FILES

`/usr/lib/calprog`	Program that figures out today's and tomorrow's dates
`/etc/passwd`	List of login directories
`/tmp/cal*`	Temporary files
`$HOME/calendar`	Personal calendar file

SEE ALSO

`mail`(1)

NAME

cat – Concatenates and prints files

SYNOPSIS

cat [-s] [-u] [-v [-t] [-e]] [*files*]

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4
AT&T extensions (-s, -v, -t, and -e options)

DESCRIPTION

The cat utility reads files you specify on the command line in sequence and writes them to standard output.

The cat utility accepts the following options and operand:

-s Makes cat silent about nonexistent files.

-u Causes the output to be unbuffered. The default is buffered output.

-v Causes nonprinting characters (except for <tab>s, <newline>s, and <form-feed>s) to be printed visibly. ASCII control characters (octal 000–037) are printed as ^*n*, where *n* is the corresponding ASCII character in the range octal 100–137 (@, A, B, C, ..., X, Y, Z, [, \,], ^, and _); the DEL character (octal 0177) is printed ^?. Other nonprintable characters are printed as M-*x*, where *x* is the ASCII character specified by the low-order 7 bits.

When used with the -v option, the following options may be used:

-t Prints tabs as ^I's and form feeds to as ^L's when used with the -v option. If you do not specify the -v option, cat ignores this option.

-e Prints a $ character at the end of each line (before the <newline> character); you must use this option with the -v option.

If the -v option is not specified, the -t and -e options are ignored.

files If you do not specify an input file, or if you specify a hyphen (-), cat reads from standard input. Multiple occurrences of the argument - are accepted as file operands.

NOTES

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
`system`, `secadm`	In a privileged administrator shell environment, shell-redirected I/O is not subject to file protections.
`sysadm`	Shell-redirected output is subject to security label restrictions.

If the `PRIV_SU` configuration option is enabled, shell-redirected I/O on behalf of the super user is not subject to file protections.

CAUTIONS

Command formats such as the following may cause the original data in *file1* to be lost:

```
$ cat file1 file2 >file1
```

EXIT STATUS

The `cat` utility exits with one of the following values:

0 All input files were output successfully.

>0 An error occurred.

EXAMPLES

Example 1: The following command line writes the contents of *file* to the standard output:

```
$ cat file
```

Example 2: The following command line concatenates the first two files and places the result in the third:

```
$ cat file1 file2 >file3
```

SEE ALSO

`cp`(1), `more`(1), `pg`(1), `pr`(1)

NAME

caterr – Processes message text files

SYNOPSIS

caterr [-c *catfile*] [-e] [-s[-P *cpp_opts*]] [-Y *x*,*pathname*] [*msgfile*]

IMPLEMENTATION

UNICOS systems

UNICOS/mk systems

IRIX systems

DESCRIPTION

A *message catalog* is a binary file that contains the run-time source of error messages output by UNICOS software products. A message catalog is produced from a message text file that contains messages (tagged with $msg tags) and message explanations (tagged with $nexp or $exp tags).

Before it can be accessed at run time, a message text file must be converted to a message catalog binary file by the caterr processor and the gencat(1) catalog generator.

The caterr utility converts the error message text source in *msgfile* into the format used as input to gencat(1), the error message catalog generation utility. If *msgfile* is not specified or if a dash (-) is specified, caterr reads from the standard input.

The -c option to the caterr utility calls gencat(1) after processing is complete. Using the -c option allows a catalog to be generated from a message text file in one step. It is recommended that you use caterr with the -c option. The gencat(1) utility exists as a separate utility to maintain compatibility with industry standards for message catalog processing. No advantage exists in calling gencat(1) separately. By default, caterr looks for gencat(1) in the /usr/bin/gencat file.

A single invocation of caterr can process either the messages or the explanations in the input files, but not both. The caterr utility processes the messages by default. Use the -e option to specify processing of the explanations.

The caterr utility calls the text formatting utility nroff(1) to process formatted explanations as part of its processing of the message text file. nroff(1) uses message macro definitions to format the explanation text. By default, on UNICOS and UNICOS/mk systems, caterr looks for nroff(1) in the /usr/bin/nroff file and for the message macros in the /usr/lib/tmac/tmac.sg file. On IRIX systems, caterr looks for nroff(1) in the /usr/bin/nroff file and for the message macros in the /usr/share/lib/tmac/tmac.sg file.

If no options are specified, caterr processes *msgfile* by using the tools in the default locations. The output, suitable for input to gencat(1), is sent to stdout.

The caterr utility accepts the following options and arguments:

-c *catfile* (Catalog) Calls gencat(1) to update or create a catalog with the information in the processed *msgfile*. If the -c option is used, caterr invokes gencat(1) to update the specified catalog by using the generated output. If *catfile* does not exist, it is created. Using the -c option makes it unnecessary to call gencat(1) separately; the message catalog is generated in one step.

-e (Explanations) Processes the explanations in *msgfile*. Without the -e option, caterr processes the messages in *msgfile*.

-s[-P *cpp_opts*]
(Symbolic names) Calls the C language preprocessor (cpp(1)) to preprocess symbolic message names into message numbers. The mapping of names to numbers must be specified in a header file name in the input file. On UNICOS and UNICOS/mk systems, caterr looks for cpp(1) first in the /usr/gen/lib/cpp directory. If it does not find it there, it looks in /lib/cpp. On IRIX systems, caterr looks for cpp(1) in the /lib/cpp directory.

Options can be passed to cpp by specifying the -P suboption to the -s option. Place the options to be passed to cpp within double quotation marks (" "). The entire string within the quotation marks is passed to cpp for execution. The -P suboption can be specified only if the -s option also is specified.

-Y *x*,*pathname*
Specifies the version of the nroff(1) and gencat(1) tools and of the tmac.sg message macros that caterr calls. If the -Y option is not specified, caterr calls the version of nroff(1) in /usr/bin/nroff, the version of gencat(1) in /usr/bin/gencat, and the version of the message macros in /usr/lib/tmac/tmac.sg (UNICOS and UNICOS/mk systems) or /usr/share/lib/tmac/tmac.sg (IRIX systems). If you need to specify alternative paths for all three tools that caterr calls, you can specify the -Y option up to three times in the same command line.

The -Y option takes two arguments: a path name and a key letter that specifies which software (nroff(1), gencat(1), or the message macros) is located at that path name. The key letter is specified first, followed by a comma (,), followed by the path name. The alternative tool path specified with *pathname* must be a full path.

The -Y option accepts the following key letters:

c Specifies that the path name following the comma is the path name for gencat(1).

m Specifies that the path name following the comma is the path name for the message macros.

n Specifies that the path name following the comma is the path name for nroff(1).

msgfile Specifies the name of the file containing the message text source to be processed.

EXAMPLES

Example 1: In the following example, `caterr` processes the messages in file `ldr.msg`. The output, sent to `stdout`, is suitable for input to `gencat`(1).

```
caterr ldr.msg
```

Example 2: In the following example, `caterr` invokes `gencat`(1) to update the messages in the `ldr.cat` catalog with the information in file `ldr.msg`.

```
caterr -c ldr.cat ldr.msg
```

Example 3: In the following example, `caterr` uses the message macros in the file `/usr/me/errmsg/tmac.sg` to produce a catalog of explanations suitable for processing by `gencat`(1). The input file is `ldr.msg`; the output is sent to `stdout`.

```
caterr -e -Y m,/usr/me/errmsg/tmac.sg ldr.msg
```

Example 4: In the following example, `caterr` uses the message macros in the current directory and invokes `gencat`(1) from `/bin/gencat` to update the explanation catalog `ldr.exp` with the information in `ldr.msg`.

```
caterr -e -c ldr.exp -Y m,tmac.sg -Y c,/bin/gencat ldr.msg
```

Example 5: In the following example, `caterr` calls `nroff` from `/usr/me/errmsg/nroff` and uses the message macros in the current directory. The input file is `ldr.msg`. Explanations suitable for processing by `gencat`(1) are output to `stdout`.

```
caterr -e -Y n,/usr/me/errmsg/nroff -Y m,tmac.sg ldr.msg
```

Example 6: In the following example, `caterr` calls alternative versions of all three tools. It uses the versions of `nroff`(1) and the message macros in the current directory, and it calls `gencat`(1) from `/bin/gencat`. Using these tools, the explanations in the `ldr.exp` file are updated with the information in the `ldr.msg` file.

```
caterr -e -c ldr.exp -Y c,/bin/gencat -Y m,tmac.sg -Y n,nroff ldr.msg
```

Example 7: In the following example, `caterr` invokes `gencat`(1) to update the messages in the `ldr.cat` catalog with the information in the `ldr.msg` file. The `caterr` utility calls `cpp`(1) to preprocess symbolic message names, and passes the `-M` option to `cpp`(1) for execution.

```
caterr -c ldr.cat -s -P "-M" ldr.msg
```

Example 8: In the following example, `caterr` invokes `gencat`(1) to update the `ldr.cat` catalog. Because no message text file name is specified, the input to `caterr` is read from the standard input.

```
caterr -c ldr.cat
```

SEE ALSO

`catxt`(1), `explain`(1), `gencat`(1), `whichcat`(1)

`catgetmsg`(3C), `catgets`(3C), `catmsgfmt`(3C), `catopen`(3C) in the *UNICOS System Libraries Reference Manual*, Cray Research publication SR–2080

`nl_types`(5), `msg`(7D) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

Cray Message System Programmer's Guide, Cray Research publication SG–2121

NAME

catxt – Extracts message explanations from a message text file

SYNOPSIS

catxt [-n *outfile*] [-s[-P *cpp_opts*]] *infile*

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The catxt utility extracts the message explanations from a message text file to ready it for printing as an error message supplement document.

Typically, the input file *infile* is the message text file that resides in the developer's program library (PL). This file usually contains the text of error messages issued to users and the text of message explanations available to users through the explain(1) command. The error messages are associated with a $msg tag in the message text file. The explanations are associated with a $nexp or $exp tag in the message text file. Printed error message supplements contain the information associated with the $nexp tags.

The catxt utility accepts the following options and arguments:

-n *outfile* Specifies the *outfile* for extracted text. To produce an error message supplement, you must extract the text associated with $nexp tags from *infile*. The catxt utility performs this extraction. If the -n option is not specified, the output is sent to stdout.

-s[-P *cpp_opts*]
(Symbolic names) Calls the C language preprocessor (cpp(1)) to preprocess symbolic message names into message numbers. The mapping of names to numbers must be specified in an include file name in the input file. catxt looks for cpp(1) first in the /usr/gen/lib/cpp directory. If it does not find it there, it looks in the /lib/cpp file.

Options can be passed to cpp(1) by specifying the -P suboption to the -s option. Place the options to be passed to cpp(1) within double quotation marks (" "). The entire string within the quotation marks is passed to cpp(1) for execution. The -P suboption can be specified only if the -s option also is specified.

infile Specifies the message text file used as input to catxt.

NOTES

In addition to extracting explanations, catxt also changes the $nexp string into the .MS (message start) macro. This macro must be present for the output file to print in the proper format.

MESSAGES

The catxt utility issues the following messages:

`No explanation number processed yet.`
: This message is issued in conjunction with several of the warnings and errors that follow in this section. It indicates that the problem reported in the associated message occurred at the beginning of the file, before any explanations were processed successfully. Use this message to locate the problem and correct it.

`Last explanation number processed is '`*num*`'.`
: This message is issued in conjunction with several of the warnings and errors that follow in this section. It indicates that the problem reported in the associated message occurred after the specified explanation number. Use this message to locate the problem and correct it.

`***WARNING*** Encountered $nexp with no explanation number.`
: Every `$nexp` tag must be followed by an explanation number. If you receive this message, there is a `$nexp` tag without a number in the input file. Begin searching for this tag after the explanation number specified in the second line of the message (one of the first two messages in this section).

`***WARNING*** Encountered $nexp followed by character(s), a number is expected.`
: Each `$nexp` tag must be followed by an explanation number. If you receive this message, a `$nexp` tag exists that is followed by characters rather than numbers. Begin searching for this tag after the explanation number specified in the second line of the message (one of the first two messages in this section).

`***WARNING*** Incorrectly formed .ME line.`
: Each explanation must end with a `.ME` macro. The `.ME` macro must be on a line by itself. If you receive this message, a `.ME` macro in the input file is followed by other characters. Begin searching for this macro after the explanation number specified in the second line of the message (one of the first two messages in this section).

`***WARNING:nexp number` *n* `does not have an ending ".ME"***`
: The `catxt` utility verifies that there is a closing `.ME` macro for each occurrence of the `$nexp` tag. If it finds a `$nexp` tag without a `.ME` macro at the end of the explanation, it issues this warning message. If you receive this warning, add the `.ME` macro to message number *n* in the input file *infile* and rerun `catxt`.

`***ERROR*** Input file` *filename* `is not in the message text file format`
: The `catxt` utility verifies that key elements of the message text file format are present in the input file. If they are not present, it issues this error message. If you receive this error message, verify that the format of the file conforms to message text file guidelines. For a detailed description of the format of the message text file, see the *Cray Message System Programmer's Guide*, Cray Research publication SG–2121.

`***ERROR*** The message text file and the output file for nroffable`

`explanations are identical! Use a different output file.`

If you receive this message, the `catxt` utility has detected that the input file (message text file) and the output file you have specified have the same name. Proceeding with the utility under these circumstances will destroy the input file. To avoid the destruction of the input file, choose a different name for the output file.

SEE ALSO

`msg`(7D) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

Cray Message System Programmer's Guide, Cray Research publication SG–2121

NAME

cb – C program beautifier

SYNOPSIS

cb [-j] [-l *length*] [-s] [*files*]

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The cb utility reads C programs either from its arguments or from the standard input and writes them on the standard output with spacing and indentation that displays the structure of the code. Under default options, cb preserves all user new lines.

The cb utility accepts the following options:

-j Causes split lines to be put back together.

-l *length* Causes cb to split lines that are longer than *length*. *length* must be between 10 and 120 characters. The default *length* is 120.

-s Changes the style of the code to conform to the style of Kernighan and Ritchie in *The C Programming Language*.

files Specifies files to read in.

CAUTIONS

The source code that is to be run under cb should be free of compilation errors.

Punctuation that is hidden in preprocessor statements causes indentation errors.

SEE ALSO

cc(1)

NAME

cd – Changes working directory

SYNOPSIS

cd [*directory*]

cd -

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The cd utility changes your working directory to *directory*. If you omit *directory*, the value of the shell variable HOME (your home directory) is used as the new working directory. If *directory* specifies a complete path starting with /, ., or .., *directory* becomes the new working directory. If neither case applies, cd tries to find the designated directory relative to one of the paths specified by the CDPATH shell variable. If CDPATH is not defined, the search path defaults to . (dot). CDPATH has the same syntax as, and similar semantics to, the PATH shell variable. cd must have execute (search) permission in *directory*.

NOTES

The cd utility is a built-in utility to the standard shell (sh(1)). An executable version of this utility is available in /usr/bin/cd.

EXIT STATUS

The cd utility exits with one of the following values:

0 The directory was successfully changed.

>0 An error occurred.

SEE ALSO

pwd(1), sh(1)

chdir(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

NAME

cdc – Changes the delta commentary of an SCCS delta

SYNOPSIS

cdc -r *SID* [-m[*mrlist*]] [-y[*comment*]] *files*

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The cdc command changes the delta commentary of the *SID* specified by the -r option of each named Source Code Control System (SCCS) file. A *delta commentary* is the modification request (MR) and comment information normally specified by using the delta(1) command (-m and -y options).

If you specify a directory, cdc behaves as though each file in the directory were specified as a named file, except that non-SCCS files (last component of the path name does not begin with s.) and unreadable files are silently ignored. If a name of - is given, the standard input is read (see the WARNINGS section); each line of the standard input is taken to be the name of an SCCS file to be processed.

Arguments to cdc, which may appear in any order, consist of option-arguments and file names.

All described option-arguments apply independently to each named file:

-r *SID* Used to specify the SCCS identification (*SID*) string of a delta for which the delta commentary is to be changed.

-m[*mrlist*] If the SCCS file has the v flag set (see admin(1)), a list of (MR) numbers to be added and/or deleted in the delta commentary of the *SID* specified by the -r option may be supplied. A null MR list has no effect.

MR entries are added to the list of MRs in the same manner as that of delta(1). To delete an MR, precede the MR number with the ! character (see the EXAMPLES section). If the MR to be deleted is currently in the list of MRs, it is removed and changed into a comment line. A list of all deleted MRs is placed in the comment section of the delta commentary and preceded by a comment line stating that they were deleted.

If -m is not used and the standard input is a terminal, the prompt MRs? is issued on the standard output before the standard input is read; if the standard input is not a terminal, no prompt is issued. The MRs? prompt always precedes the comments? prompt (see -y option).

MRs in a list are separated by blanks and/or tab characters. An unescaped new-line character terminates the MR list.

If the `v` flag has a value (see `admin`(1)), it is taken to be the name of a program (or shell procedure) that validates the correctness of the MR numbers. If a nonzero exit status is returned from the MR number validation program, `cdc` terminates and the delta commentary remains unchanged.

`-y`[*comment*] Arbitrary text used to replace the *comment*(s) already existing for the delta specified by the `-r` option.

files Files to be changed.

The previous comments are kept and preceded by a comment line stating that they were changed. A null *comment* has no effect.

If `-y` is not specified and the standard input is a terminal, the prompt `comments?` is issued on the standard output before the standard input is read; if the standard input is not a terminal, no prompt is issued. An unescaped newline character terminates the *comment* text.

Permissions needed to modify the SCCS file are either (1) if you made the delta, you can change its delta commentary or (2) if you own the file and directory, you can modify the delta commentary.

WARNINGS

If SCCS file names are supplied to the `cdc` command through the standard input (`-` on the command line), the `-m` and `-y` options must also be used.

MESSAGES

Error messages from SCCS are printed. Use `help`(1) for explanations.

EXAMPLES

The following example adds `bl78-12345` and `bl79-00001` to the MR list, removes `bl77-54321` from the MR list, and adds the comment `trouble` to delta `1.6` of `s.file`:

```
$ cdc -r1.6 -m"bl78-12345 !bl77-54321 bl79-00001" -ytrouble s.file
```

The following example does the same thing:

```
$ cdc -r1.6 s.file
MRs? !bl77-54321 bl78-12345 bl79-00001
comments? trouble
```

FILES

`x.`*file* See `delta`(1)

`z.`*file* See `delta`(1)

SEE ALSO

admin(1), comb(1), delta(1), get(1), help(1), prs(1), rmdel(1), sact(1), sccsdiff(1), unget(1), val(1), vc(1), what(1)

sccsfile(5) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

NAME

cflow – Generates C-language flowgraph

SYNOPSIS

cflow [-d *num*] [-D *name*[=*def*]]... [-i *incl*] [-I *dir*]... [-r] [-U *dir*]... *file*...

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The cflow utility analyzes a collection of C, yacc, lex, assembler, and object files and builds a graph charting the external function references. Files suffixed with .y, .l, and .c are processed by yacc, lex, and the C compiler as appropriate. The results of the preprocessed files, and files suffixed with .i, are then run through the first pass of lint. Files suffixed with .s are assembled. Assembled files, and files suffixed with .o, have information extracted from their symbol tables. The results are collected and turned into a graph of external references that is written on the standard output.

Each line of output begins with a reference number, followed by a suitable number of tabs indicating the level, then the name of the global symbol followed by a colon and its definition. Normally only function names that do not begin with an underscore are listed (see the -i options). For information extracted from C source, the definition consists of an abstract type declaration (for example, char *), and, delimited by angle brackets, the name of the source file and the line number where the definition was found. Definitions extracted from object files indicate the file name and location counter under which the symbol appeared (for example, *text*). Leading underscores in C-style external names are deleted.

Once a definition of a name has been printed, subsequent references to that name contain only the reference number of the line where the definition may be found. For undefined references, only <> is printed.

As an example, suppose the following code is in file.c:

```
int   i;

main()
{
        f();
        g();
        f();
}

f()
{
        i = h();
}
```

The command

```
$ cflow -ix file.c
```

produces the output

```
1       main: int(), <file.c 4>
2               f: int(), <file.c 11>
3                       h: <>
4                       i: int, <file.c 1>
5               g: <>
```

When the nesting level becomes too deep, the output of `cflow` can be piped to the `pr`(1) utility, using the `-e` option, to compress the tab expansion to something less than every eight spaces.

In addition to the `-D`, `-I`, and `-U` options (which are interpreted just as they are by `cc`(1)), the following options are interpreted by `cflow`:

`-r` Reverses the ‘‘caller:callee’’ relationship producing an inverted listing showing the callers of each function. The listing is also sorted in lexicographical order by callee.

`-ix` Includes external and static data symbols. The default is to include only functions in the flowgraph.

`-i_` Includes names that begin with an underscore. The default is to exclude these functions (and data if `-ix` is used).

`-d` *num* The *num* decimal integer indicates the depth at which the flowgraph is cut off. By default this number is very large. Attempts to set the cutoff depth to a nonpositive integer will be ignored.

NOTES

Files produced by `lex` and `yacc` cause the reordering of line number declarations, which can confuse `cflow`. To get proper results, feed `cflow` the `yacc` or `lex` input.

DIAGNOSTICS

Complains about multiple definitions and only believes the first.

SEE ALSO

`cc`(1), `lex`(1), `lint`(1), `nm`(1), `pr`(1), `yacc`(1)

NAME

`chacid` – Changes account ID of disk files

SYNOPSIS

`chacid` [`-v`] [`-S`] [`-h`] *files*
`chacid` [`-s` *id*] [`-v`] [`-h`] *files*

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `chacid` utility allows users to set the account ID of a disk file (or symbolic link). The ID must be one of the user's valid accounts. Only the owner of a file may change that file's account ID. An appropriately authorized user may set the account ID of a file that is owned by another user and may specify any account ID value.

When used without options, `chacid` displays the current account ID. Options `-s` and `-S` are mutually exclusive.

The `chacid` utility accepts the following options and arguments:

`-s` *id* Sets the account ID of the specified *files* to *id*.

`-v` Operates in verbose mode.

`-S` Sets the account ID of files to the current user's default account ID.

`-h` When this option is specified and the file is a symbolic link, the requested operation (display the current value or change the account ID) is done on the link; that is, it does not follow the link to the destination file. If this option is not specified, the link is followed and the operation is done on the destination file.

files Specifies the file (or files) whose account ID will be set.

NOTES

If this utility is installed with a privilege assignment list (PAL), a user who is assigned the following privilege text upon execution of this command is allowed to perform the actions shown:

Privilege Text	Action
`anyown`	Allowed to change the account ID of a file that is owned by another user and to specify any account ID value.
`own`	Allowed to change the account ID of a file that is owned by another user.
`any`	Allowed to specify any account ID value.

If this utility is installed with a PAL, a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
`system`, `secadm`	Allowed to change the account ID of any file to any value.
`sysadm`	Allowed to change the account ID of any file to any value, subject to security label restrictions on the file's path. Allowed to specify any account ID value.

If the `PRIV_SU` configuration option is enabled, the super user is allowed to change the account ID of any file to any value.

EXAMPLES

To determine the current account ID of the file `notes`, enter the following:

```
chacid notes
```

To change the account ID of the file `notes` to `proj1`, enter the following:

```
chacid -s proj1 notes
```

FILES

`/etc/acid`	Account ID information file that contains account names and account IDs
`/etc/udb`	User validation file that contains user control limits

SEE ALSO

`newacct`(1), `privtext`(1)

`chacid`(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

UNICOS Resource Administration, Cray Research publication SG–2302

NAME

checknr – Checks nroff and troff input files; reports possible errors

SYNOPSIS

checknr [-fs] [-a .*x1* .*y1* .*x2* .*y2* ... .*xn* .*yn*] [-c .*x1* .*x2* .*x3* ... .*xn*] [*filename* ...]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The checknr utility checks a list of nroff(1) or troff(1) input files for certain kinds of errors involving mismatched opening and closing delimiters and unknown commands. If no files are specified, checknr checks the standard input. Delimiters checked are as follows:

- Font changes using \f*x* ... \fP.
- Size changes using \s*x* ... \s0.
- Macros that come in open ... close forms (for example, the .TS and .TE macros), which must always come in pairs.

The checknr utility knows about the ms(7D) and me(7D) macro packages.

The checknr utility is intended to be used on documents that are prepared with checknr in mind. It expects a certain document writing style for \f and \s commands, in that each \f*x* must be terminated with \fP and each \s*x* must be terminated with \s0. While it will work to go directly into the next font or explicitly specify the original font or point size, and many existing documents actually do this, such a practice will produce complaints from checknr. Since it is probably better to use the \fP and \s0 forms anyway, you should think of this as a contribution to your document preparation style.

The checknr utility accepts the following options:

-f	Ignores \f font changes.
-s	Ignores \s size changes.
-a .*x1* .*y1* ...	Adds pairs of macros to the list. The pairs of macros are assumed to be those (such as .DS and .DE) that should be checked for balance. The -a option must be followed by groups of six characters, each group defining a pair of macros. The six characters are a period, the first macro name, another period, and the second macro name. For example, to define the pair .BS and .ES, use -a.BS.ES.
-c .*x1* ...	Defines commands that checknr would otherwise complain about as undefined.
filename	nroff file to be checked.

NOTES

There is no way to define a 1-character macro name using the `-a` option.

SEE ALSO

`eqn`(1), `nroff`(1), `troff`(1)

`me`(7D), `ms`(7D) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

NAME

chgrp – Changes the group ownership of a file

SYNOPSIS

chgrp [-R] [-h] *group file* ...

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4
AT&T extensions (-h option)

DESCRIPTION

The chgrp utility changes the group ID of each *file* to *group*.

The chgrp utility accepts the following options and operands:

-R Recursively changes file group IDs. When symbolic links are encountered, the group of the target is changed, but no recursion takes place.

-h If the file is a symbolic link, changes the group of the symbolic link. Without this option, the group of the file or directory referenced by the symbolic link is changed.

group May be either a decimal group ID or a group name found in the group file.

file ... Files or directories to be changed. For each *file* operand that specifies a directory, chgrp changes the group ID of the directory and all files in the file hierarchy below it. When symbolic links are encountered, they are not traversed.

Only an appropriately authorized user may change the group of a file that is owned by another user. Unless users are appropriately authorized, they must be a member of the specified group to change the group of a file.

Unless the user is appropriately authorized, chgrp clears the set-user-ID and set-group-ID file mode bits.

NOTES

When the path name supplied to the chgrp utility specifies a multilevel symbolic link (the name of a multilevel directory), the group is changed only on the root of the multilevel directory tree. While this affects all subsequently created labeled subdirectories, it does not affect labeled subdirectories currently in existence. To ensure that the group of all labeled subdirectories change, the user must manually change the group on each labeled subdirectory found in root of the multilevel directory.

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
`system`, `secadm`	Allowed to change the group of any file to any value. The set-user-ID and set-group-ID file mode bits are not cleared.
`sysadm`	Allowed to change the group of any file to any value. The set-user-ID and set-group-ID file mode bits are not cleared. Shell-redirected I/O is subject to security label restrictions.

If the `PRIV_SU` configuration option is enabled, the super user is allowed to change the group of any file to any value. If the user is the super user or has the `suidgid` permission, the set-user-ID and set-group-ID file mode bits are not cleared.

EXIT STATUS

The `chgrp` utility exits with one of the following values:

0 All requested changes were made.

>0 An error occurred.

EXAMPLES

The following example changes the group to `dev` for the `example.c` file:

```
chgrp dev example.c
```

FILES

`/etc/udb`	User validation file that contains user control limits
`/etc/group`	Group file that contains group names and group IDs

SEE ALSO

`chmod`(1), `chown`(1), `id`(1)

`chown`(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

`group`(5), `passwd`(5), `udb`(5) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

General UNICOS System Administration, Cray Research publication SG–2301

NAME

chkey – Changes your encryption key

SYNOPSIS

```
chkey
```

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The chkey command prompts users for their secure RPC password, and uses it to encrypt a new encryption key for the user to be stored in the publickey(5) database.

SEE ALSO

keylogin(1)

publickey(5) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

keyserv(8), newkey(8) in the *UNICOS Administrator Commands Reference Manual*, Cray Research publication SR–2022

NAME

chkpnt – Checkpoints a process, multitask group, or job

SYNOPSIS

```
chkpnt -j id [-k] [-v] [-f file]
chkpnt -p id [-k] [-v] [-f file]
```

IMPLEMENTATION

Cray PVP systems

DESCRIPTION

The chkpnt utility creates a restart file containing all of the necessary state information to restart all of the processes selected by the *id* option. The target *id* must belong to the current user, unless the current user is the super user.

The chkpnt utility accepts the following options:

`-j` *id*
`-p` *id* Indicates whether the *id* specified is a job (or interactive session) ID (`-j`) or a process ID (`-p`).

`-k` Indicates that the specified *id* (job or process) will be killed if the checkpoint is successful.

`-v` Causes additional informational messages to be printed.

`-f` *file* Indicates the path name to be used for the restart file. If the user does not specify a path name, a default path name will be created by appending *id* to the name `pid` or `jib`.

NOTES

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
system, secadm, sysadm	Allowed to checkpoint any process or job.

If the PRIV_SU configuration option is enabled, the super user is allowed to checkpoint any process or job.

The following restrictions apply to jobs and processes that are to be checkpointed:

- Only an appropriately authorized user may checkpoint a process or job that is owned by another user.
- The active security label of the user must equal the active security label of every affected process.
- Processes with open pipes may be checkpointed and restarted successfully if the following two conditions are met:
  - All openings of the pipe file must be contained within the process collection being checkpointed.

  - All I/O operations on the pipe must be atomic with respect to the chkpnt system call. This condition is a limit on the size of an I/O operation: either PIPE_BUF bytes, or (v_maxpipe * 4096) bytes. PIPE_BUF is found in the sys/param.h file. v_maxpipe is a member of the var structure in the sys/var.h file.

- All files that a process was using when it was checkpointed must be present when the process is restarted.
- Processes using shared memory segments (CRAY T90 series systems only) cannot be checkpointed or restarted.
- Processes using online tapes cannot be checkpointed or restarted.

EXAMPLES

The following example illustrates how to use the chkpnt utility to checkpoint a process.

The user is running a.out in the background and enters a chkpnt utility to checkpoint the process. The -p option specifies the PID and the -k option kills the process (pid.23634). The user then issues an ls command to list information about pid.23634; the capital R at the beginning of the long listing indicates a restart file has been created.

```
unicos$ a.out &
23634

unicos$ chkpnt  -p  23634  -k

unicos$ ls  -l  pid.23634
Rr--------  1  (id) (group) (size) (date) pid.23634
```

Later, the user enters a restart utility to recover the process. The ps listing confirms the process has been reactivated.

```
unicos$ restart  pid.23634

unicos$ ps
    PID    TTY  TIME COMMAND
  23634   p031  0:13 a.out
  23510   p031  0:00 sh
  23758   p031  0:00 ps
```

SEE ALSO

chkpnt_util(1), chkptint(1), restart(1)

chkpnt(2), restart(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

General UNICOS System Administration, Cray Research publication SG–2301

NAME

chkpnt_util – Verifies restartability of restart files

SYNOPSIS

chkpnt_util [-f *filch*] [-n] [-o *name*] [-v] *restart-file*

IMPLEMENTATION

Cray PVP systems

DESCRIPTION

The chkpnt_util utility verifies a restart file in the same manner that the kernel would verify the file during an attempted restart. The utility issues the same cryptic messages as the kernel, but offers a verbose mode that locates the problem. The utility also enables you to find the name of a restart-referenced nc1 or sfs file.

The chkpnt_util accepts the following options and operand:

-f *filch* Associates path names to the files identified by specified vnodes. Items in the *filch* string are separated by commas. Each item is a pair separated by a colon. The first in the pair is the former vnode pointer, used to identify the node within the restart file. The second in the pair is the new file name.

-n Prints the name of every file referenced in the restart file. This option uses a lot of execution time.

-o *name* Writes a new restart file with the specified name.

-v Produces verbose output.

restart-file Specifies the restart file.

NOTES

Only an appropriately authorized user may provide super-user authority to write a restart file. Similarly, authority is required for the openi(2) system call.

EXAMPLES

The following example shows the output produced when you specify `chkpnt_util` with the `-n` and `-v` options:

```
# chkpnt_util -n -v  ofile
chkpnt_util: restart header
chkpnt_util: session header, sid = 63
chkpnt_util: process executing sh
chkpnt_util:   process pcomm structure, ppid 3392
chkpnt_util:     task proc structure, pid 3395
chkpnt_util:   process ucomm structure, uc_magic 020001600407
chkpnt_util:       vnode for executable text
chkpnt_util:         vnode structure from vp = 01251570
chkpnt_util:         -- inum 5305 on minor 48
chkpnt_util:         -- file name "/bin/sh"
chkpnt_util:       vnode for current directory
chkpnt_util:         vnode structure from vp = 01526754
chkpnt_util:         -- inum 106 on minor 36
chkpnt_util:         -- file name "/tmp/jtmp.000057a"
chkpnt_util:       open file number 0
chkpnt_util:         file structure, from fp = 01204253
chkpnt_util:         vnode structure from vp = 01254320
chkpnt_util:         -- inum 1530 on minor 48
chkpnt_util:         -- file name "/dev/tty"
....
chkpnt_util:       open file number 31
chkpnt_util:         file structure, from fp = 01204551
chkpnt_util:         vnode structure from vp = 01536764
chkpnt_util:         -- inum 1061919 on minor 27
chkpnt_util:         -- file name "/frost/u4/rig/.sh_history"
chkpnt_util: file has been modified
chkpnt_util: **** condition prevents restart
```

To obtain detailed information about specific files, select the vnodes and associated new file names from the verbose output and enter the following command:

```
chkpnt_util -f 01526754:/tmp/jtmp.000057b, 1536764:/dev/null
```

SEE ALSO

`chkpnt`(1), `restart`(1)

`chkpnt`(2), `openi`(2), `restart`(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

General UNICOS System Administration, Cray Research publication SG–2301

NAME

chkptint – Registers current session to be checkpointed upon shutdown and/or periodically

SYNOPSIS

chkptint -s *sec*

IMPLEMENTATION

Cray PVP systems

DESCRIPTION

The chkptint utility notifies the Unified Resource Manager (URM) that a checkpoint of the current session should be taken every *sec* seconds and/or at shutdown, depending on the URM configuration. This utility should be used in conjunction with the rmgr(1) utility View Restart subcommand, which allows you to see if you have any checkpointed interactive sessions.

The purpose of the URM checkpoint feature is to improve user recoverability by allowing you to periodically checkpoint your active session to protect against lost work due to an uncontrolled system shutdown and to request that an interactive session be checkpointed as part of a controlled system shutdown.

The chkptint utility requires the following options:

-s *sec* A positive integer *sec* indicates that a checkpoint of the current session should be done at shutdown and the frequency at which periodic checkpoints should be taken. Zero *sec* indicates that checkpointing should not be done for the current session.

The units of the -s option are either wall-clock or CPU seconds; the unit selected is specified in a URM configuration parameter.

NOTES

The URM checkpoint facility is disabled by default; checkpointing of sessions is not performed unless this facility is enabled in URM. A URM configuration parameter, chkpt_switch, can be set to select whether checkpointing is done at shutdown only or both periodically and at shutdown. If URM is running with checkpoint at shutdown only, a nonzero value for *sec* also indicates that checkpointing is desired at shutdown. For information on enabling checkpointing in URM, see *UNICOS Resource Administration*, Cray Research publication SG–2302.

The checkpointing of sessions through chkptint follows all the current limitations of chkpnt(2). For a list of restrictions, see chkpnt(2).

If you restart a saved session, this session does not show up in the output of the finger(1B) or who(1) utility. The reason is that init(8) is the process that updates the tables used by these commands and restarting a previously running session does not notify the init(8) process.

MESSAGES

```
chkptint:  Could not obtain the current session ID
using the getjtab system call.

chkptint:  The current session ID returned by the getjtab system call
does not match the job table data.

chkptint:  The current user ID returned by the getjtab system call
does not match the job table data.
```

EXAMPLES

To set a checkpoint interval frequency of 30 minutes:

```
$ chkptint -s 1800
```

To automatically turn off checkpointing:

```
$ chkptint -s 0
```

SEE ALSO

chkpnt(1), finger(1B), restart(1), rmgr(1), who(1)

chkpnt(2), getjtab(2), restart(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

init(8) in the *UNICOS Administrator Commands Reference Manual*, Cray Research publication SR–2022

UNICOS Resource Administration, Cray Research publication SG–2302

NAME

chmod – Changes mode of files or directories

SYNOPSIS

chmod [-R] *mode files*

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The chmod utility changes the mode of the specified files (or directories) according to *mode*.

The chmod utility accepts the following option and operands:

-R Recursively descends through directory arguments, setting the mode for each file. When symbolic links are encountered, the mode of the target is changed, but no recursion occurs.

mode Specifies the permissions and other attributes of a file. The mode may be absolute or symbolic.

files Files to be changed.

Absolute changes to modes are stated with octal numbers:

chmod *nnnn files*

n is a number from 0 through 7.

An absolute mode is given as an octal number constructed from the OR of the following modes:

Code	Description
4000	Sets user ID on execution.
20#0	Sets group ID on execution if # is 7, 5, 3, or 1. Enables mandatory locking if # is 6, 4, 2, or 0. (Secure sites: see secure site information that follows.) If the file is a directory, this bit is ignored; you may set or clear it using only the symbolic mode.
1000	Sticky bit (applicable only to directories). See chmod(2) for more information.
0400	Sets read permission for owner.
0200	Sets write permission for owner.
0100	Sets execute (search in directory) permission for owner.
0070	Sets read, write, and execute (search in directory) permission for group.

`0007` Sets read, write, and execute (search in directory) permission for others.

Symbolic changes are stated with mnemonic characters:

`chmod [`*who*`] `*operator*` [`*permissions(s)*`] `*files*

who is one or more characters that corresponds to user, group, or other (`u`, `g`, or `o`, respectively); *operator* is +, –, or =, signifying the assignment of permissions; and *permission(s)* is one or more characters that correspond to type of permission.

Symbolic changes are stated using letters that correspond both to access classes and to the individual permissions themselves. Permissions to a file may vary, depending on your user identification number (UID) or group identification number (GID). Permissions are described in three sequences, each having three characters:

User	Group	Other
rwx	rwx	rwx

The preceding lines (meaning that user, group, and others all have reading, writing, and execution permission to a given file) demonstrate two categories for granting permissions: the access class and the permissions themselves.

A command making *file* readable and writable by the group using the symbolic method would appear as follows:

`chmod g+rw `*file*

who can be stated as one or more of the following letters:

`u` User's permissions

`g` Group's permissions

`o` Others permissions

`a` All (equivalent to specifying `ugo`).

If you omit *who*, `chmod` will use the file creation mask (see `umask`) to determine whose permissions are to be changed.

operator is one of `+`, `-`, or `=`, signifying how permissions are to be changed:

`+` Adds permissions.

`-` Removes permissions.

`=` Assigns permissions absolutely.

Unlike other symbolic operations, = has an absolute effect in that it resets all other bits. Omitting *permission(s)* is useful only with = to remove all permissions.

permission(s) is any compatible combination of the following letters:

r Reading permission

w Writing permission

x Execution permission

X Search/execute permission if the file is a directory or if current mode has at least one of the execute bits set.

s User or group set-ID is turned on

l Mandatory locking occurs during access (if any execution bits are set, group set-ID is turned on).

t Sets the sticky bit (see chmod(2) for more information)

Multiple symbolic modes separated by commas can be specified, although no spaces may intervene between these modes. Operations are performed in the order given. Multiple symbolic letters following a single operator cause the corresponding operations to be performed simultaneously. The letter s is meaningful only with u or g; thus, if a is used with s, only u and g are affected. t works only with u.

Mandatory file and record locking refers to a file's ability to have its read or write permissions locked while a program is accessing that file. You cannot permit group execution and enable a file to be locked on execution at the same time. You can turn on the set-group-ID and enable a file to be locked on execution at the same time.

The following examples are, therefore, illegal usages and will elicit error messages:

```
chmod g+x,+l file

chmod g+s,+l file
```

Only an appropriately authorized user may change the mode of a file that he or she does not own. Only an appropriately authorized user may alter the t permission (mode 1000). Otherwise, chmod clears the t permission, but does not return an error. To enable a file's set-group-ID mode bit, you must belong to the file's owning group and group execution must be allowed.

NOTES

The chmod utility permits you to produce useless modes if they are not illegal (for example, making a text file executable).

When the path name supplied to the chmod utility specifies a multilevel symbolic link (the name of a multilevel directory), the mode is changed only on the root of the multilevel directory tree. While this affects all subsequently created labeled subdirectories, it does not affect labeled subdirectories currently in existence. To ensure that the mode of all labeled subdirectories change, the user must manually change the mode on each labeled subdirectory found in root of the multilevel directory.

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
`system`, `secadm`	Allowed to change the mode of any file. The set-user-ID and set-group-ID file mode bits are not cleared.
`sysadm`	Allowed to change the mode of any file. The set-user-ID and set-group-ID file mode bits are not cleared. Shell-redirected I/O is subject to security label restrictions.

If the `PRIV_SU` configuration option is enabled, the super user is allowed to change the mode of any file. If the user is the super user or has the `suidgid` permission, the set-user-ID and set-group-ID file mode bits are not cleared.

EXIT STATUS

The `chmod` utility exits with one of the following values:

0 All requested changes were made.

>0 An error occurred.

EXAMPLES

Example 1: The following examples show how to deny execution permission to all. The absolute (octal) example permits only reading permissions:

```
chmod a-x file
chmod 444 file
```

Example 2: The following examples make a file readable and writable by the group and others:

```
chmod go+rw file
chmod 066 file
```

Example 3: The following example locks a file during access:

```
chmod +l file
```

Example 4: The following examples enable all to read, write, and execute the file; and they turn on the set-group-ID:

```
chmod +rwx,g+s file
chmod 2777 file
```

SEE ALSO

`ls`(1), `umask`(1)

`chmod`(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

NAME

chown – Changes owner of files or directories

SYNOPSIS

chown [-h] [-R] *owner*[:*group*] *files...*

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4
AT&T extensions (-h option)

DESCRIPTION

The chown utility changes the owner of each *file* to *owner*. The optional *group* operand changes the group.

The chown utility accepts the following options and operands:

-h If the file is a symbolic link, changes the owner of the symbolic link. Without this option, the owner of the file or directory referenced by the symbolic link is changed.

-R Recursively changes file user IDs, and if the *group* operand is specified, group IDs. When symbolic links are encountered, the owner of the target is changed, but no recursion takes place. For each *file* operand that specifies a directory, chown changes the user and group ID of the directory and all files in the file hierarchy below it.

owner[:*group*]
owner may be either a decimal user ID or a login name found in the password file. *group* may be either a decimal group ID or a group name found in the group file.

files... Files or directories to be changed.

Only an appropriately authorized user may change the owner of a file.

Unless the user is appropriately authorized, chown clears the set-user-ID and set-group-ID file mode bits.

NOTES

When the path name supplied to the chown utility specifies a multilevel symbolic link (the name of a multilevel directory), the owner is changed only on the root of the multilevel directory tree. While this affects all subsequently created labeled subdirectories, it does not affect labeled subdirectories currently in existence. To ensure that the owner of all labeled subdirectories change, the user must manually change the owner on each labeled subdirectory found in root of the multilevel directory.

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
system, secadm	Allowed to change the owner of any file. The set-user-ID and set-group-ID file mode bits are not cleared.
sysadm	Allowed to change the owner of any file. The set-user-ID and set-group-ID file mode bits are not cleared. Shell-redirected I/O is subject to security label restrictions.

If the PRIV_SU configuration option is enabled, the super user is allowed to change the owner of any file. If the user is the super user or has the suidgid permission, the set-user-ID and set-group-ID file mode bits are not cleared. A user with the chown permbit is allowed to change the ownership of his or her files.

EXIT STATUS

The chown utility exits with one of the following values:

0 All requested changes were made.

>0 An error occurred.

FILES

/etc/udb	User validation file that contains user control limits
/etc/passwd	Password file that contains login names and user IDs
/etc/group	Group file that contains group names and group IDs

SEE ALSO

chgrp(1), chmod(1)

chown(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

group(5), passwd(5), udb(5) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

udbgen(8) in the *UNICOS Administrator Commands Reference Manual*, Cray Research publication SR–2022

General UNICOS System Administration, Cray Research publication SG–2301

NAME

chsh – Changes default login shell

SYNOPSIS

/usr/ucb/chsh *name* [*shell*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The chsh utility is similar to passwd(1) except that it changes the login shell field of your password file rather than the password entry.

The *name* argument is your login name.

The *shell* argument is compared with the list of legal user shells obtained from the getusershell(3C) library routine. Possible shells, depending on site configuration, may be one of the following:

```
/bin/sh
/bin/csh
```

The full path name is not necessary. Only an appropriately authorized user can specify a shell that is not in the allowed list of user shells or change the shell for another user. If you do not specify a shell, the login shell defaults to /bin/sh.

NOTES

If this utility is installed with a privilege assignment list (PAL), a user who is assigned the following privilege text upon execution of this command is allowed to perform the action shown:

Privilege Text	Action
chgany	Allowed to change the shell of any user to any value.

If this utility is installed with a PAL, a user with one of the following active categories is allowed to perform the action shown:

Active Category	Action
system, secadm, sysadm	Allowed to change the shell of any user to any value.

If the PRIV_SU configuration option is enabled, the super user is allowed to change the shell of any user to any value.

EXAMPLES

```
chsh bill /bin/csh
chsh jim /bin/sh
```

SEE ALSO

csh(1), passwd(1), privtext(1), sh(1)

getusershell(3C) in the *UNICOS System Libraries Reference Manual*, Cray Research publication SR–2080

shells(5), udb(5) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

General UNICOS System Administration, Cray Research publication SG–2301

NAME

cksum – Writes file checksums and sizes

SYNOPSIS

cksum [*files...*]

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The cksum utility calculates and writes to standard output a cyclic redundancy check (CRC) for each input file, and it also writes to standard output the number of octets in each file. The CRC used is based on the polynomial used for CRC error checking in the networking standard ISO 8802-3. The standard input is used only if no *file* operands are specified.

For each file processed successfully, the cksum utility writes in the following format:

"<*checksum*> <*# of octets*> <*path name*>\n"

If you omit the *file* operand, the path name and its leading space is omitted.

The cksum utility accepts the following operand:

files... A path name of a file (any type) to be checked. If no *file* operands are specified, the standard input is used.

NOTES

The generating polynomial defines the encoding, as follows:

$$G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$$

Mathematically, the CRC value that corresponds to a given file is defined by the following procedure:

1. The *n* bits to be evaluated are considered to be the coefficients of a mod 2 polynomial *M(x)* of degree *n*-1. These *n* bits are the bits from the file, with the most-significant bit being the most significant bit of the first octet of the file and the last bit being the least-significant bit of the last octet, padded with zero bits (if necessary) to achieve an integral number of octets, followed by one or more octets representing the length of the file as a binary value, least-significant octet first. The smallest number of octets that can represent this integer is used.
2. *M(x)* is multiplied by x^{32} (for example, shifted left 32 bits) and divided by *G(x)* using mod 2 division, producing a remainder *R(x)* of degree ≤ 31.

3. The coefficients of *R(x)* are considered to be a 32-bit sequence.
4. The bit sequence is complemented, and the result is the CRC.

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
`system`, `secadm`	Allowed to generate a checksum of any file. In a privileged administrator shell environment, shell-redirected I/O is not subject to file protections.
`sysadm`	Allowed to generate a checksum of any file subject to security label restrictions. Shell-redirected I/O is subject to security label restrictions.

If the `PRIV_SU` configuration option is enabled, the super user is allowed to generate a checksum of any file. Shell-redirected I/O on behalf of the super user is not subject to file protections.

EXAMPLES

Generate a checksum for each of the files `a.out`, `file.c`, and `obj.o`:

```
cksum a.out file.c obj.o
```

EXIT STATUS

The `cksum` utility exits with one of the following values:

0 All files were processed successfully.

>0 An error occurred.

SEE ALSO

ISO 8802-3: 1989, *Information processing systems – Local area networks – Part 3: Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specification.*

A Tutorial on CRC Computations, Ramabadran and Gaitonde, *IEEE Micro*, August, 1988, p. 62.

Computation of Cyclic Redundancy Checks Via Table Lookup, Sarwate, Dilip V, *Communications of the ACM*, August, 1988, p. 1011.

NAME

`clear` – Clears terminal screen

SYNOPSIS

`/usr/ucb/clear`

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `clear` command clears your screen. It looks in the environment for the terminal type and then in `/usr/lib/terminfo` to determine how to clear the screen.

FILES

`/usr/lib/terminfo`	Terminal capability database

NOTES

Use the `tput clear` command, because the `clear` command might be removed in a future UNICOS 9.0 release.

SEE ALSO

`tput`(1)

NAME

cmp – Compares two files

SYNOPSIS

```
cmp [-l] file1 file2
cmp -s file1 file2
cmp -w file1 file2
```

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4
CRI extension (-w option)

DESCRIPTION

The cmp utility compares *file1* and *file2*. Under default options, cmp makes no comment if the files are the same; if they differ, cmp displays the byte and line number at which the difference occurs. If one file is an initial subsequence of the other, that fact is noted.

The cmp utility accepts the following options:

-l Displays the byte number (decimal) and the differing bytes (octal) in three columns for each difference.

-s Displays nothing for files that differ; returns exit status only.

-w Word mode. Displays differences between 64-bit words. The octal offset of the word from the beginning of the file, an octal representation of the words, and an ASCII representation of the words are displayed for each word that differs.

file1
file2 The path names of the two files to be compared. If - is specified, the standard input is used.

NOTES

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
system, secadm	In a privileged administrator shell environment, shell-redirected I/O is not subject to file protections.
sysadm	Shell-redirected output is subject to security label restrictions.

If the PRIV_SU configuration option is enabled, shell-redirected I/O on behalf of the super user is not subject to file protections.

EXIT STATUS

The cmp utility exits with one of the following values:

0 Files are identical.

1 Files are different.

2 An error occurred.

SEE ALSO

comm(1), diff(1)

NAME

col – Filters reverse line feeds

SYNOPSIS

col [-b] [-f] [-p] [-x]

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The col utility filters reverse line feeds, reading from the standard input and writing to the standard output. It performs the line overlays implied by reverse line feeds (ASCII code ESC-7), and by forward and reverse half-line feeds (ESC-9 and ESC-8). col is particularly useful for filtering multicolumn output made with the .rt command of nroff(1) and output resulting from use of the tbl(1) preprocessor.

The col utility accepts the following options:

-b Indicates that the output device is incapable of backspacing.

-f Enables half-line vertical motion.

-p Enables output of escape sequences as regular characters.

-x Suppresses output of tabs rather than white space.

If the -b option is given, col assumes that the output device in use is incapable of backspacing. In this case, if 2 or more characters are to appear in the same place, only the last one read will be output.

Although col accepts half-line motions in its input, it normally does not emit them on output. Instead, text that would appear between lines is moved to the next lower full-line boundary. This treatment can be suppressed by the -f (fine) option; in this case, the output from col may contain forward half-line feeds (ESC-9), but will still never contain either kind of reverse line motion.

The ASCII control characters SO (\017) and SI (\016) are assumed by col to start and end text in an alternate character set. The character set to which each input character belongs is remembered; on output SI and SO characters are generated as appropriate to ensure that each character is printed in the correct character set.

On input, the only control characters accepted are <space>, <backspace>, <tab>, <carriage-return>, <newline>, SI, SO, <vertical tab> (\013), and ESC followed by 7, 8, or 9. The <vertical tab> character is an alternative form of full reverse line feed, included for compatibility with some earlier programs of this type. All other nonprinting characters are ignored.

Normally, `col` ignores any unknown escape sequences found in its input; the `-p` option may be used to cause `col` to output these sequences as regular characters, subject to overprinting from reverse line motions. The use of this option is highly discouraged unless you are fully aware of the textual position of the escape sequences.

Unless the `-x` option is given, `col` will convert white space to tabs on output wherever possible to shorten printing time.

NOTES

The input format accepted by `col` matches the output produced by `nroff`(1) with either the `-T37` or `-Tlp` options. Use `-T37` (and the `-f` option of `col`) if the ultimate disposition of the output of `col` will be a device that can interpret half-line motions; otherwise, use `-Tlp`.

EXIT STATUS

The following exit values are returned:

0 successful completion.

>0 An error occurred.

BUGS

The following are known `col` bugs:

- The `col` utility cannot back up more than 128 lines.
- Up to 800 characters, including backspaces, are allowed on a line.
- Local vertical motions that would result in backing up over the first line of the document are ignored. As a result, the first line must not have any superscripts.

SEE ALSO

`nroff`(1), `tbl`(1)

NAME

comb – Combines SCCS deltas

SYNOPSIS

comb [-c*list*] [-o] [-p*sid*] [-s] *files*

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The comb command generates a shell procedure (see sh(1)) which, when run, reconstructs the given Source Code Control System (SCCS) files.

The reconstructed files should be smaller than the original files. The options may be specified in any order, but all options apply to all named SCCS files. If a directory is named, comb behaves as though each file in the directory were specified as a named file, except that non-SCCS files (last component of the path name does not begin with s.) and unreadable files are silently ignored. If a dash (-) is given as a name, the standard input is read; each line of the input is taken to be the name of an SCCS file to be processed; non-SCCS files and unreadable files are silently ignored. If no options are specified, comb will preserve only leaf deltas and the minimal number of ancestors needed to preserve the tree.

The generated shell procedure is written on the standard output.

The options are as follows. Each is explained as though only one named file is to be processed, but the effects of any option apply independently to each named file.

-c*list* Prints a *list* (see get(1) for the syntax of a *list*) of deltas to be preserved. All other deltas are discarded.

-o Causes the reconstructed file to be accessed at the release of the delta to be created for each get -e generated. Otherwise, the reconstructed file would be accessed at the most recent ancestor. Use of the -o option may decrease the size of the reconstructed SCCS file. It may also alter the shape of the delta tree of the original file.

-p*sid* Provides the SCCS identification string (*sid*) of the oldest delta to be preserved. All older deltas are discarded in the reconstructed file.

-s Causes comb to generate a shell procedure which, when run, produces a report. This report gives for each file: the file name, size (in blocks) after combining, original size (also in blocks), and percentage change computed by the following:

100*(*original–combined*)/*original*

You should use this option to determine exactly how much space is saved by the combining process before any SCCS files are actually combined.

file1 file2 Files to be compared.

MESSAGES

Error messages from SCCS are printed. Use `help`(1) for explanations.

BUGS

The `comb` command may rearrange the shape of the tree of deltas. It may not save any space; in fact, it is possible for the reconstructed file to actually be larger than the original.

The `comb` command may produce shell scripts that contain `admin` commands which have an argument to the `-fi` option containing white space. This will cause the shell script to fail. To allow the shell script to work, place single quotes around the argument to the `-fi` option.

FILES

`s.COMB`	Name of the reconstructed SCCS file
`comb`*?????*	Temporary file

SEE ALSO

`admin`(1), `cdc`(1), `delta`(1), `get`(1), `help`(1), `prs`(1) `rmdel`(1), `sact`(1), `sccsdiff`(1), `sh`(1), `unget`(1), `val`(1), `vc`(1), `what`(1)

`sccsfile`(5) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

NAME

comm – Selects or rejects lines common to two sorted files

SYNOPSIS

comm [-1] [-2] [-3] *file1 file2*

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The comm utility reads *file1* and *file2*, which should be ordered in ASCII collating sequence (see sort(1)), and produces a three-column output: lines only in *file1*; lines only in *file2*; and lines in both files. File name - means that standard input is used.

The comm utility accepts the following options:

-1
-2
-3 Suppresses printing of the corresponding column. Thus, comm -12 prints only the lines common to the two files, comm -23 prints only lines in the first file but not in the second, and comm -123 prints nothing.

NOTES

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
system, secadm	Allowed to manage any sorted files. In a privileged administrator shell environment, shell-redirected I/O is not subject to file protections.
sysadm	Allowed to manage any sorted files subject to security label restrictions. Shell-redirected I/O is subject to security label restrictions.

If the PRIV_SU configuration option is enabled, the super user is allowed to manage any sorted files. Shell-redirected I/O on behalf of the super user is not subject to file protections.

EXIT STATUS

The comm utility exits with one of the following values:

0 All input files were successfully output as specified.

>0 An error occurred.

SEE ALSO

cmp(1), diff(1), sort(1), uniq(1)

NAME

command – Executes a simple command

SYNOPSIS

command [-p] *command_name* [*argument...*]
command [-v] *command_name*
command [-V] *command_name*

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The command utility causes the shell to treat the arguments as a simple command, suppressing the shell function lookup (see sh(1)).

In every other respect, if *command_name* is not the name of a shell function, the effect of command is the same as omitting command.

The command utility accepts the following options and operands:

-p Performs the command search using a default value for PATH that will find all of the standard utilities.

-v Writes a string to standard output that indicates the path name or command that the shell will use in the current shell execution environment, to invoke *command_name*.

-V Writes a string to standard output that indicates how the shell will interpret the name given in the *command_name* operand in the current shell execution environment.

argument A string treated as an argument to *command_name*.

command_name The name of a utility or a special built-in utility.

NOTES

The command utility described in this man page is a built-in utility to the standard shell (sh(1)). An executable version of this utility is available in /usr/bin/command.

EXIT STATUS

The command utility exits with one of the following values:

0 Successful completion.

>0 The *command_name* cannot be found or an error occurred.

126 The utility specified by *command_name* was found but could not be invoked.

127 An error occurred in the command utility or the utility specified by *command_name* could not be found.

Otherwise, the exit status of command is that of the simple command specified by the arguments to command.

SEE ALSO

sh(1)

NAME

compress – Compresses expanded files

SYNOPSIS

compress [-c] [-f] [-v] [-b *bits*] [*filename*]

compress [-f] [-v] [-b *bits*] [*filename* ...]

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4
AT&T

DESCRIPTION

The compress utility reduces the size of the named files using adaptive Lempel-Ziv coding. A .Z extension is added to the compressed file name. The ownership modes, access time, and modification time stay the same. If no files are specified, the standard input is compressed to the standard output.

The amount of compression obtained depends on the size of the input, the number of *bits* per code, and the distribution of common substrings. Usually, text such as source code or English is reduced by 50 to 60%. Compression is generally much better than that achieved by Huffman coding (as used in pack(1)), and takes less time to compute. The *bits* parameter specified during compression is encoded within the compressed file, along with a magic number to ensure that neither decompression of random data nor recompression of compressed data is subsequently allowed.

Compressed files can be restored to their original form using the uncompress utility.

The compress utility accepts the following options:

-b *bits* Sets the upper limit (in bits) for common substring codes. *bits* must be between 9 and 16 (16 is the default). Lowering the number of bits will result in larger, less-compressed files. For a portable application, bits must be between 9 and 14, inclusive.

-c Writes to the standard output; no files are changed. The nondestructive behavior of zcat is identical to that of specifying uncompress -c.

-f Forces compression, even if the file does not actually shrink, or the corresponding .Z file already exists. When running in the background (under /bin/sh), and -f is not specified, prompt to verify whether an existing .Z file should be overwritten.

-v Verbose. Displays the percentage reduction for each file compressed.

filename File to be compressed.

NOTES

Although compressed files are compatible between machines with large memory, `-b 12` should be used for file transfer to architectures with a small process data space (64 Kbytes or less).

The `compress` utility should be more flexible about the existence of the `.Z` suffix.

EXIT STATUS

The following exit values are returned:

0 Successful completion.

1 An error occurred.

2 One or more files were not compressed because they would have increased in size (and the `-f` options was not specified).

>2 An error occurred.

MESSAGES

`Usage: compress` [`-fvc`] [`-b maxbits`] [*filename* ...]
Invalid options were specified on the command line.

`Missing maxbits`
Maxbits must follow `-b`.

filename`: not in compressed format`
The file specified to `uncompress` has not been compressed.

filename`: compressed with` *xx*`bits, can only handle` *yy*`bits`
filename was compressed by a program that could deal with more *bits* than the compress code on this machine. Recompress the file with smaller *bits*.

filename`: already has .Z suffix -- no change`
The file is assumed to be already compressed. Rename the file and try again.

filename`: already exists; do you wish to overwrite (y or n)?`
Respond `y` if you want the output file to be replaced; `n` if not.

`Compression:` *xx.xx*%
Percentage of the input saved by compression. (Relevant only for `-v`.)

`-- not a regular file: unchanged`
When the input file is not a regular file, (such as a directory), it is left unaltered.

`-- has` *xx* `other links: unchanged`
The input file has links; it is left unchanged. See `ln`(1) for more information.

`-- file unchanged`
No savings are achieved by compression. The input remains uncompressed.

SEE ALSO

pack(1), sh(1), uncompress(1), zcat(1)

''A Technique for High Performance Data Compression,'' Terry A. Welch, *IEEE Computer*, vol. 17, no. 6 (June 1984), pp. 8–19.

NAME

cp – Copies files

SYNOPSIS

cp [-f] [-i] [-p] [-r] [-R] *file1* [*file2* ...] *target*

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The cp utility copies the contents of a file to another path name, or copies the contents of one or more files into path names in another directory.

The cp utility accepts the following options:

-f If an existing file cannot be overwritten, unlinks the existing file and re-creates it.

-i Prompts for confirmation whenever the copy overwrites an existing *target*. To proceed with the copy, specify y. Any other answer prevents cp from overwriting *target*.

-p Duplicates not only the contents of *file*, but also preserves the modification time and permission modes. The access control list (ACL) is preserved also.

-r, -R If *file* is a directory, cp will copy the directory and all its files, including any subdirectories and their files; *target* must be a directory.

files A path name of a file (any type) to be checked. If no *file* operands are specified, the standard input is used.

target A path name of a target file where the files will be copied.

The cp utility copies *file* to *target*. *file* and *target* may not have the same name. (Care must be taken when using shell metacharacters.) If *target* is not a directory, only one file may be specified before it; if *target* is a directory, more than one file may be specified. If *target* does not exist, cp creates a file named *target*. If *target* exists and is not a directory, its contents are overwritten. If *target* is a directory, the file(s) are copied to that directory.

If *file* is a directory, *target* must be a directory in the same physical file system. *target* and *file* do not have to share the same parent directory.

If *file* is a file and *target* is a link to another file with links, the other links remain and *target* becomes a new file.

If *target* does not exist, `cp` creates a new file named *target*, which has the same mode as *file* except that the sticky bit is not set unless the user is a privileged user; the owner and group of *target* are those of the user.

If *target* is a file, its contents are overwritten, but the mode, owner, and group associated with it are not changed. The last modification time of *target* and the last access time of *file* are set to the time the copy was made.

If *target* is a directory, a new file with the same mode is created in the target directory for each file named; the file's user ID is set to the effective user ID of the process, and the file's group ID is set to the group ID of the directory in which the file is copied.

NOTES

A `--` permits users to mark the end of any command line options explicitly, thus allowing `cp` to recognize file name arguments that begin with a `-`. If a `--` and a `-` both appear on the same command line, the second is interpreted as a file name.

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	**Action**
`system`, `secadm`	Allowed to copy any file. The values of set-user-ID and set-group-ID mode bits are preserved.
`sysadm`	Allowed to copy any file, subject to security label restrictions on the source and destination file paths. The values of set-user-ID and set-group-ID mode bits are preserved. Shell-redirected I/O is subject to security label restrictions.

If the `PRIV_SU` configuration option is enabled, the super user is allowed to copy any file. The values of set-user-ID and set-group-ID mode bits are preserved.

EXIT STATUS

The `cp` utility exits with one of the following values:

0 No error occurred.

>0 An error occurred.

WARNINGS

Beware of a recursive copy that copies the contents of a directory to one of its subdirectories. For example:

```
$ cp -r ~/src ~/src/bkup
```

will keep copying files until it fills the entire file system.

EXAMPLES

Example 1: Use the following command line to copy file `copyone` to `copytwo`, which already exists:

```
$ cp -i copyone copytwo
overwrite copytwo? y
$
```

Example 2: Use the following command line to copy all files in your working directory that begin with the letter `b` to subdirectory `copieshere`:

```
$ cp b* copieshere
```

Example 3: Use the following command line to copy file `file.c` to subdirectory `newdir`. The copy will have the name `copy.c`.

```
$ cp file.c newdir/copy.c
```

Example 4: Use the following command line to make a copy of the directory `dir1`, naming it `dirnew`:

```
$ cp -r dir1 dirnew
```

SEE ALSO

`cat`(1), `chmod`(1), `cpio`(1), `ln`(1), `mv`(1), `rcp`(1), `rm`(1)

`chmod`(2), `creat`(2), `open`(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

General UNICOS System Administration, Cray Research publication SG–2301

NAME

cpio – Copies file archives in and out

SYNOPSIS

cpio -o [-a] [-c] [-e] [-v] [-x] [-z [-x] [-M] [-P]] [-B]

cpio -i [-6] [-c] [-d] [-h *hdr-type*] [-f] [-m] [-r] [-t] [-u] [-v] [-x] [-z [-x] [-M] [-P]] [-B] [-E *filename*] [*patterns*]

cpio -p [-a] [-d] [-e] [-l] [-m] [-u] [-v] [-x] [-z [-x] [-M] [-P]] *directory*

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The cpio utility, invoked with the -o option, reads the standard input to obtain a list of path names and copies those files onto the standard output together with path name and status information.

Output is padded to a 512-byte boundary. Using the -z option outputs security information, access control lists (ACLs), and privilege assignment lists (PALs) corresponding to those files.

The -i option extracts files from the standard input, which is assumed to be the product of a previous cpio -o. Only files with names that match *patterns* are selected. *Patterns* are given in the name-generating notation of sh(1). In *patterns*, metacharacters ?, *, and [...] match the slash (/) character. Use double quotation marks ("), single quotation marks(’), or backslashes (\) to protect the metacharacters from being expanded by the shell. Multiple *patterns* may be specified and if no *patterns* are specified, the default for *patterns* is * (that is, select all files). The extracted files are conditionally created and copied into the current directory tree based upon the options described in the following. The permissions of the files will be those of the previous cpio -o. The owner and group of the files will be that of the current user, unless the user is appropriately authorized to make cpio retain the owner and group of the files of the previous cpio -o; see the NOTES section. When reading an input created with the -z option, specifying the -z option on input preserves security information, ACLs, and PALs.

The -p option reads the standard input to obtain a list of path names of files that are conditionally created and copied into the destination directory tree *directory* based upon the options described in the following. Using the -z option preserves security information, ACLs, and PALs to copied files.

Only files with security levels and compartments that are dominated by the user’s active security levels and compartments may be copied. Appropriately authorized users can copy any file; see the NOTES section.

Only authorized users can copy special files; see the NOTES section.

On UNICOS systems using multilevel security labels, any user can archive data to a single-level medium. Only appropriately authorized users can archive data to a multilevel medium. Data restored from any medium is assigned the original ACL, no PAL, and the mandatory access control security attributes of the invoking user. An appropriately authorized user can restore the original file attributes and PAL by using the `-M` option. Only appropriately authorized user can restore data from a multilevel medium; see the NOTES section.

The available options are as follows. A dash (`-`) may precede the following options; dashes may be omitted if the options are concatenated without spaces.

`-6` Processes a UNIX System Sixth Edition format file. Useful only with the `-i` option (copy in).

`-a` Resets access times of input files after they have been copied.

`-B` Blocks input/output at 5120 bytes to the record (does not apply to the `-p` option; useful only with data that will be stored on tape).

`-c` Writes header information in ASCII character form for portability. This option is necessary to transfer a `cpio` archive between a Cray Research mainframe and another machine. Only an appropriately authorized user can copy restart files by using this option; see the NOTES section. This option is not valid when using the (`-oz`) option.

`-d` Creates directories as needed.

`-e` Verifies all parameters of the `stat` structure (`stat`(2)) are within certain limits. If any value is invalid, the file is not copied into the archive. This option is not guaranteed to exist in future UNICOS releases. By default, a file is copied regardless of its `stat` structure contents.

`-E` *filename*
Specifies an input file that contains a list of file names to be extracted from the archive (one file name per line).

`-f` Copies in all files except those in *patterns*.

`-h` *hdr-type*
By default, `cpio` attempts to automatically interpret the `cpio` non-ASCII header that precedes each file in the `cpio` archive (see `cpio`(5)). The `-h` option allows control over the interpretation of this non-ASCII header. If *hdr-type* is `o`, `cpio` interprets the headers as having been created in a pre-6.0 UNICOS release. If *hdr-type* is `n`, `cpio` interprets the headers as having been created in a UNICOS 6.0 or later release. If *hdr-type* is incorrectly chosen, a `phase error` will most likely occur and `cpio` will not read the archive. Use this option if `cpio` does not automatically interpret the non-ASCII header correctly. Use only with the `-i` option.

`-l` Whenever possible, link files rather than copying them. Use only with the `-p` option.

`-m` Retains previous file modification time. This is ineffective on directories that are being copied.

`-M` Preserves all security attributes of all files. Useful only with the `-z` option.

`-P` Excludes copy or preservation of PALs. Useful only with the `-z` option.

`-r` Interactively renames files. The user will be prompted with the name of the file. If the user types a null line, the file is skipped. If the line consists of a single period, the file is processed with no modification to its name. Otherwise, it's name will be replaced with the contents of the line. Use only with the `-i` option.

`-t` Prints table of contents of the input. No files are created.

`-u` Copies unconditionally. (Usually, an older file does not replace a newer file with the same name).

`-v` (Verbose) Causes a list of file names to be printed. When used with the `-t` option, the table of contents looks like the output of an `ls -l` command (see `ls`(1)).

`-x` Excludes copy or preservation of ACLs. Useful only with the `-z` option.

`-z` Copies security information and ACLs.

NOTES

The `-x`, `-M`, `-P` and `-z` options are valid only when used with the `-z` option.

Files generated with the `-z` option can be processed only using the `-z` option.

Appropriately authorized users are users that are assigned the following privilege text:

Privilege Text	Action Allowed
`LKSU`	Retains owner/group of a file and copies special files or restart files
`UPIP`	Restores security attributes from unnamed pipe
`ALL`	Restores security attributes from any medium
`ALLLKSU`	Combined behavior of `ALL` and `LKSU` privilege texts
`UPIPLKSU`	Combined behavior of `UPIP` and `LKSU` privilege texts

Note that the `LKSU` privilege text is the only one used in the PAL supplied by default. The `UPIP`, `ALL`, `ALLLKSU`, and `UPIPLKSU` privilege texts are supported if your site wants to use site-defined PALs.

If the `PRIV_SU` configuration option is enabled, root is an appropriately authorized user.

BUGS

Path names are restricted to 256 characters. If there are too many unique linked files, the program runs out of memory to keep track of them and, thereafter, linking information is lost.

EXAMPLES

Example 1: This example copies the contents of a directory into an archive.

```
ls | cpio -o >/archive/file
```

Example 2: This example duplicates a directory hierarchy.

```
cd olddir
find . -depth -print | cpio -pdl newdir
```

Example 3: The following example uses the online tape subsystem to write and read `cpio` tapes:

```
rsv
tpmnt -l nl -p /tmp/tapedev -v vsn -b 4096
find . -print | cpio -cov > /tmp/tapedev
cd newdirectory
dd if=/tmp/tapedev bs=4096 | cpio -civd
rls -a
```

SEE ALSO

`ar`(1), `cp`(1), `find`(1), `ls`(1), `pax`(1), `privtext`(1), `rls`(1), `rsv`(1), `spset`(1), `tar`(1), `tpmnt`(1)

`chown`(2), `getpal`(2), `getppriv`(2), `setdevs`(2), `setpal`(2), `setppriv`(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

`acl`(5), `cpio`(5) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

Tape Subsystem User's Guide, Cray Research publication SG–2051

UNICOS Resource Administration, Cray Research publication SG–2302

NAME

crontab – Creates or modifies the user's crontab file

SYNOPSIS

```
crontab [file]
crontab -e
crontab -l
crontab -r
```

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The crontab utility creates, replaces, or edits a user's crontab entry, which is a list of commands and the times at which they are to be executed. To input the new crontab entry, specify *file* on the command line. If you omit *file*, standard input is used. Use the -e to invoke an editor for the crontab file.

The crontab utility accepts the following options and operands:

-e Edits a copy of the invoking user's crontab entry, or creates an empty entry to edit if the crontab entry does not exist. When editing is complete, the entry is installed as the user's crontab entry.

-l Lists the invoking user's crontab entry.

-r Removes a user's crontab entry from the crontab directory.

file Name of the file that contains the crontab entry.

Users who are permitted to use crontab are those whose names appear in the /usr/lib/cron/cron.allow file. If that file does not exist, the /usr/lib/cron/cron.deny file is checked to determine whether a user should be denied access to crontab. If neither file exists, only root is allowed to submit a job. The null file cron.allow would mean that no users are allowed to use cron; null file cron.deny would mean that no users are denied the use of cron. The allow/deny files consist of one user name per line.

A crontab file consists of lines of six fields each. The fields are separated by <space>s or <tab>s. The first five are integer patterns that specify the following:

minute (0-59)
hour (0-23)
day of the month (1-31)
month of the year (1-12)
day of the week (0-6 with 0=Sunday)

Each of these patterns may be either an asterisk (meaning all legal values) or a list of elements separated by commas. An element is either one number or two numbers separated by a minus sign (meaning an inclusive range). The specification of days may be made by two fields (day of the month and day of the week). If you specify both as a list of elements, both are followed. For example, `0 0 1,15 * 1` would run a command on the first and fifteenth of each month, as well as on every Monday. To specify days by only one field, the other field should be set to * (for example, `0 0 * * 1` would run a command only on Mondays).

The sixth field of a line in a `crontab` file is a string that is executed by the shell at the specified times. A percent character (%) in this field (unless escaped by \) is translated as a `<newline>` character. The shell executes only the first line (up to a % or end-of-line character) of the command field. The other lines are made available to the command as standard input.

The shell is invoked from your `$HOME` directory with an `arg0` of `sh`. Users who want to have their `.profile` file executed must so state that explicitly in the `crontab` file. `cron`(8) supplies a default environment for every shell, defining `HOME`, `LOGNAME`, `SHELL(=/bin/sh)`, and `PATH(=:/bin:/usr/bin:/usr/ucb)`. You will probably want to set the `TZ` environment variable.

Blank lines and lines that begin with # are ignored.

At the time of submission, `crontab` files are run at the user's current security label.

NOTES

Redirect the standard output and standard error files of commands; otherwise, all generated output and errors will be mailed to you.

If this utility is installed with a privilege assignment list (PAL), a user who is assigned the following privilege text upon execution of this command is allowed to perform the action shown:

Privilege Text	Action
`showall`	Allowed to manage all jobs.

If this utility is installed with a PAL, a user with one of the following active categories is allowed to perform the action shown:

Active Category	Action
`system`, `secadm`, `sysadm`, `sysops`	Allowed to manage all jobs.

If the `PRIV_SU` configuration option is enabled, the super user is allowed to manage all jobs.

EXIT STATUS

The `crontab` utility exits with one of the following values:

0 Successful completion.

>0 An error occurred.

EXAMPLES

Example 1: The following contents appear in a file in the `/usr/lib/cron/crontabs` directory. The `/u/tom/bin/hskp` file is executed by the shell at 3:00 A.M. every day of the month, every month of the year, Tuesday through Saturday:

```
0  3  *  *  2-6  /u/tom/bin/hskp
```

Example 2: The following contents appear in a file in the `/usr/lib/cron/crontabs` directory. The file `$HOME/diskwatch` is executed by the shell every half-hour, Monday through Friday:

```
0,30  *  *  *  1-5  $HOME/diskwatch
```

FILES

`/usr/lib/cron`	Main `cron` directory
`/usr/lib/cron/cron.allow`	List of allowed users
`/usr/spool/cron/crontabs`	Spool area
`/usr/lib/cron/cron.deny`	List of denied users
`/usr/lib/cron/log`	Accounting information

SEE ALSO

`sh`(1)

`queuedefs`(5) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

`cron`(8) in the *UNICOS Administrator Commands Reference Manual*, Cray Research publication SR–2022

NAME

crypt – Encodes or decodes a file

SYNOPSIS

```
crypt [-O] [password]
crypt [-k] [-O]
```

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The crypt utility reads from standard input and writes on standard output. If the *password* argument is not specified, crypt demands a key from the terminal and turns off printing while the key is being typed. If the -k option is used, crypt uses the key assigned to the environment variable CrYpTkEy.

The crypt utility accepts the following options:

-O Decrypts files previously encrypted on a UNICOS system before release 5.0.

password Specifies a key that selects a particular transformation.

-k Causes crypt to use the key assigned to the CrYpTkEy environment variable.

The crypt utility encrypts and decrypts with the same key:

```
crypt key <clear >cypher
crypt key <cypher | pr
```

Files encrypted by crypt are compatible with those treated by the ed(1), edit(1), ex(1), and vi(1) editors in encryption mode.

The security of encrypted files depends on three factors: 1) the fundamental method must be hard to solve; 2) direct search of the key space must be infeasible; 3) ‘‘sneak paths’’ by which keys or clear text can become visible must be minimized.

The crypt utility implements a one-rotor machine designed along the lines of the German Enigma, but with a 256-element rotor. Methods of attack on such machines are known, but not widely; moreover, the amount of work required is likely to be large.

The transformation of a key into the internal settings of the machine is deliberately designed to be expensive, that is, to take a substantial fraction of a second to compute. However, if keys are restricted to three lowercase letters, encrypted files can be read in only a substantial fraction of 5 minutes of machine time.

If the key is an argument to the crypt utility, it is potentially visible to users executing ps(1) or a derivative. To minimize this possibility, crypt takes care to destroy any record of the key immediately upon entry. The choices of keys and key security are the most vulnerable aspect of crypt.

NOTES

The inclusion of encryption code requires a special license for sites outside the United States. If these encryption functions are not available on your system, check with your system support staff.

BUGS

If output is piped to `nroff` and the encryption key is not specified on the command line, `crypt` can leave terminal modes in a strange state (see `stty`(1)).

EXAMPLES

Example 1: The following example encrypts file `secrets` into file `secure`. A key must be provided (note that printing is turned off while the key is being entered). User input is shown in bold type:

```
$ crypt < secrets > secure
Enter key:
```

Example 2: To print the file to `stdout`, enter the following by using the same key as you did to encrypt the previous file:

```
$ crypt < secure
Enter key:
This is the text of the file secrets.
```

FILES

`/dev/tty`	Controlling input terminal for typed key

SEE ALSO

`ed`(1), `edit`(1), `ex`(1), `makekey`(1), `ps`(1), `stty`(1), `vi`(1)

NAME

csh – Invokes the C shell

SYNOPSIS

csh [-b] [-c] [-e] [-f] [-i] [-n] [-s] [-t] [-v] [-x] [-S] [-V] [-X] [*args*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The C shell is a command interpreter that has a syntax similar to that of the C language. The csh utility incorporates a history mechanism (see the History Substitution subsection) and job control facilities.

A session with csh begins with the execution of commands from file .cshrc in your home directory. However, if this is a login shell, csh executes commands from file /etc/cshrc first, then .cshrc and .login in your home directory.

The shell begins reading commands from the terminal after the % prompt; it then repeatedly performs the following actions:

1. Reads a line of command input
2. Breaks the line of command input into *words* (described under Lexical Structure)
3. Puts the sequence of words in the command history list (described under History Substitution)
4. Parses the command history list
5. Executes each command on the current line

When a login shell terminates, csh executes commands from the .logout file in the user's home directory.

If argument 0 to the shell is a dash (-), this is a login shell.

The csh utility accepts the following options:

-b Forces a *break* from option processing. Subsequent command-line arguments are not interpreted as C shell options. This allows the passing of options to a script without confusion. The shell does not run a set-user-ID or set-group-ID script unless this option is present.

-c Reads a single command or file of commands from *args*. Any remaining arguments are placed in the *argv* variable. If no argument is specified, this option will have no effect.

-e Exits when an invoked command terminates abnormally or yields a nonzero exit status.

-f Suppresses startup processing, which consists of searching for and executing the .cshrc file in your home directory. This makes the transition into the shell faster.

`-i` Specifies an interactive shell and prompts for its top-level input, even if it appears to not be a terminal. Shells are interactive without this option when their inputs and outputs are terminals.

`-n` Parses but does not execute commands. This helps check for accuracy in the syntax of shell scripts.

`-s` Takes command input from the standard input.

`-t` Reads and executes one line of input. Use a backslash (\) to escape the new-line character at the end of this line and continue to another line.

`-v` Sets the `verbose` variable, so that command input is echoed after history substitution.

`-x` Sets the `echo` variable, so that commands are echoed immediately before execution.

`-S` Sets the `timestamp` variable. Prefixes commands with a date and time stamp in the form *day month date hh:mm:ss.*

`-V` Sets the `verbose` variable even before `.cshrc` is executed.

`-X` Sets the `echo` variable even before .`cshrc` is executed.

After argument processing, if arguments (*args*) remain but you did not specify the `-c`, `-i`, `-s`, or `-t` option, the first argument is taken as the name of a file of commands to be executed. The shell opens this file and saves its name for possible resubstitution by `$0`. Remaining arguments initialize the *argv* variable.

Lexical Structure

The shell splits input lines into words at blanks and tabs with the following exceptions. The `&`, `|`, `;`, `<`, `>`, `(`, and `)` characters form separate words. If the `&`, `|`, `<`, or `>` characters are doubled to `&&`, `||`, `<<`, or `>>`, respectively, the pairs form single words. You can use these parser metacharacters as part of other words or override their special meaning by preceding them with \. (A new line preceded by \ is equivalent to a blank.)

In addition, strings enclosed in matched pairs of quotation marks ' or "), form parts of a word; metacharacters in these strings, including blanks and tabs, do not form separate words. Quotation mark semantics are described in the Quotations with ' and " subsection. Within pairs of ' or " characters, a newline preceded by a \ produces a true newline character.

When the shell's input is not a terminal, the # character introduces a comment, which continues to the end of the input line. To override this special meaning, precede the # with a \ and use ´, `, and " in quotation marks.

Commands

A simple command is a sequence of words, the first of which specifies the command you want to execute. A simple command or a sequence of simple commands separated by `|` characters forms a pipeline. The output of each command in a pipeline is connected to the input of the next. You can separate sequences of pipelines with a semicolon (`;`). The piped commands are then executed sequentially. You can execute a sequence of pipelines without waiting for the sequence to terminate by following it with an ampersand (&).

You can put any of the preceding characters in parentheses to form a simple command (which can be a component of a pipeline). You can also separate pipelines with || or &&, indicating, as in the C language, that the second is to be executed only if the first fails or succeeds, respectively. (See the Expressions subsection.)

Built-in commands

The csh utility accepts the following list of built-in commands (execution of nonbuilt-in commands is described later). When a built-in command occurs as any component of a pipeline except the last, it is executed in a subshell.

alias
alias *name*
alias *name wordlist*

The first form prints all aliases. The second form prints the alias for *name*. The final form assigns the specified *wordlist* as the alias of *name; wordlist* is command and file name substituted. *name* cannot be alias or unalias. The C shell restricts the number of nested alias substitutions on a line to 20.

bg
bg %*job*...

Runs the current or specified jobs in the background.

break

Causes execution to resume after the end of the nearest enclosing foreach or while. The remaining commands on the current line are executed. You can thus produce multilevel breaks by writing them all on one line.

breaksw

Causes a break from switch, resuming after endsw.

case *label:*

Labels a switch statement, as discussed under the default command.

cd
cd *name*
chdir
chdir *name*

Changes the shell's working directory to directory *name*. If no argument is specified, it changes to the home directory of the user. If *name* is not found as a subdirectory of the current directory (and does not begin with /, ./, or ../), each component of the variable cdpath is checked to see whether it has a subdirectory *name*. Finally, if all else fails but *name* is a shell variable with a value that begins with /, then this is tried to see whether it is a directory.

continue

Continues the execution of the nearest enclosing while or foreach. The rest of the commands on the current line are executed.

default:

Labels the default case in a switch statement. The default should come after all case labels.

dirs

Prints the directory stack. The top of the stack is at the left; the first directory in the stack is the current directory.

dmmode

`dmmode` *n* Sets the data migration recall mode to *n*. See `dmmode`(1) for usage and description.

`echo` *wordlist*
`echo -n` *wordlist*
Writes the specified words to the shell's standard output, separated by spaces and terminated with a new-line character unless the `-n` option is specified.

`else`
`end`
`endif`
`endsw` See the description of the `foreach`, `if`, `switch`, and `while` statements.

`eval`,*arg* ... As in `sh`(1), the arguments are read as input to the shell and the resulting commands are executed in the context of the current shell. This is usually used to execute commands generated as the result of command or variable substitution, because parsing occurs before these substitutions.

`exec` *command* Executes the specified command in place of the current shell.

`exit`
`exit` *(expr)* Exits the shell with either the value of the *status* variable (first form) or the value of the specified *expr* (second form).

`fg`
`fg` *%job* ... Brings the current or specified jobs into the foreground, continuing them if they were stopped.

`foreach` *name (wordlist)*
.
.
.
`end` Successively sets the variable *name* to each member of *wordlist* and executes the sequence of commands between this command and the matching `end`. (Both `foreach` and `end` must appear alone on separate lines.)

The `continue` statement is used to continue the loop prematurely and `break` to terminate it prematurely. When this command is read from the terminal, the loop is read once, prompting with `?` before any statements in the loop are executed. If you make a mistake typing in a loop at the terminal, you can delete it.

`glob` *wordlist* Similar to `echo`, but words are delimited by null characters in the output. `glob` is useful for programs that use the shell to file-name-expand a list of words.

`goto` *word* The specified *word* is file-name-expanded and command-expanded to yield a string of the form `label`. The shell rewinds its input as much as possible and searches for a line of the form `label`: possibly preceded by blanks or tabs. Execution continues after the specified line.

`history`
`history` *n*
`history -r` *n*
`history -h` *n* Displays the history event list. If *n* is specified, only the *n* most recent events are printed. The `-r` option reverses the order of printout so that the most recent is first instead of the oldest first. The `-h` option causes the history list to be printed without leading numbers. This is used to produce files suitable to the `source` command, using the `-h` option to `source`.

`if` (*expr*) *command*
If the specified expression evaluates true, the single *command* with arguments is executed. Variable substitution (see the Variable Substitution subsection) on *command* happens simultaneously with the rest of the `if` command. *command* must be a simple command (not a pipeline), a command list, or a parenthesized command list. Pipelines and command lists are executed outside the `if` command. I/O redirection occurs when *command* is not executed, even if *expr* is false (this is a bug).

`if` (*expr*) `then`
.
.
.
`else if` (*expr2*) `then`
.
`else`
.
`endif` If the specified *expr* is true, the commands up to the first `else` are executed; if *expr2* is true, the commands up to the second `else` are executed, and so on. Any number of `else-if` pairs are possible; only one `endif` is needed. The `else` part is likewise optional. (The words `else` and `endif` must appear at the beginning of input lines; the `if` must appear alone on its input line or after an `else`; `if`, `then`, `else`, and `endif` must be separate words.)

`jobs`
`jobs -l` Lists the active jobs. The `-l` option lists the process IDs in addition to the normal information.

`kill` %*job*
`kill` -*sig* %*job* ...
`kill` *id*
`kill` -*sig pid* ...
`kill -l` Sends either the `SIGTERM` (terminate) signal or the specified signal to the specified jobs or processes. Signals are given either by number or by names (as given in `/usr/include/signal.h`, stripped of the prefix `SIG`). The signal names are listed by `kill -l`. There is no default; `kill` alone does not send a signal to the current job.

`logout` Terminates a login shell. Especially useful if the `ignoreeof` variable is set.

`nice`
`nice +`*number*
`nice` *command*
`nice +`*number*`command`

The first form adds 4 to the current `nice` value for this shell. The second form adds *number* to the current `nice` value. The final two forms run *command* at priority 4 plus the current `nice` value, and *number* plus the current `nice` value, respectively. Super users may specify negative niceness by using `nice -`*number* .... The *command* is always executed in a subshell, and the restrictions placed on commands in simple `if` statements apply. The system imposes a maximum `nice` value of 39 and a minimum `nice` value of 0.

`nohup` When used in shell scripts, causes hangups to be ignored for the remainder of the script. All processes detached with an ampersand (&) can effectively use `nohup`.

`notify`
`notify` %*job* ...

Causes the shell to notify the user asynchronously when the status of the current or specified job changes; usually notification is presented before a prompt. This is automatic if the shell variable `notify` is set.

`onintr`
`onintr -`
`onintr` *label* Controls the action of the shell on interrupts. The first form restores the default action of the shell on interrupts, which is to terminate shell scripts and return to the terminal command input level. The second form, `onintr -`, causes all interrupts to be ignored. The final form causes the shell to execute a `goto label` when it receives an interrupt or when a child process terminates because it was interrupted.

If the shell is running detached and interrupts are being ignored, none of the forms of `onintr` have meaning, and interrupts continue to be ignored by the shell and all invoked commands.

`popd`
`popd +`*n* Pops the directory stack, returning to the new top directory. With the argument +*n*, `popd` discards the *n*th entry in the stack. The elements of the directory stack are numbered from 0, starting at the top.

`pushd`
`pushd` *name*
`pushd +`*n* The first form, without arguments, exchanges the top two elements of the directory stack. The second form changes to the new directory (as does `cd`) and pushes the old current working directory onto the directory stack. The last form, `pushd` with a numeric argument, rotates the *n*th argument of the directory stack so that it is the top element and changes it. The members of the directory stack are numbered from the top, starting at 0.

`rehash` Recomputes the internal hash table of the contents of the directories in the path variable. This is needed if new commands are added to directories in the path while you are logged in. This should be necessary only if you add commands to one of your own directories, or if a systems programmer changes the contents of one of the system directories.

`repeat` *count command*
Executes the specified command, which is subject to the same restrictions as *command* in the one-line `if` statement, *count* times. I/O redirections occur once, even if *count* is 0.

`set`
`set` *name*
`set` *name=word*
`set` *name[index]=word*
`set` *name=(wordlist)*
The first form of the command shows the value of all shell variables. Variables that have a value other than a single word print as a parenthesized word list. The second form sets *name* to the null string. The third form sets *name* to the single *word.* The fourth form sets the *index* component of *name* to *word*; this component must already exist. The final form sets *name* equal to the list of words in *wordlist*. In all cases, the value is command and file-name-expanded.

These arguments may be repeated to set multiple values in a single `set` command. Note however, that variable expansion is done for all arguments before any setting occurs.

The *name* argument can be a maximum of 18 characters long and must begin with a letter. (The underscore character is considered a letter.) When entering the command line, include either one or more spaces on both sides of the `=` sign or no space on either side.

You can repeat these arguments to set multiple values in a single `set` command. Variable expansion is done for all arguments before any setting occurs.

`setenv`
`setenv` *name value*
`setenv` *name* The first form lists all current environment variables. The second form sets the value of environment variable *name* to *value,* a single string. The last form sets *name* to an empty string. The most commonly used environment variables (`USER`, `TERM`, and `PATH`) are automatically imported to and exported from the `csh` variables `user`, `term`, and `path`; there is no need to use `setenv` for these.

The *name* argument can be a maximum of 18 characters long and must begin with a letter. The underscore character is considered a letter.

`setucat` *cat* Sets the active category. See `setucat`(1) for usage and description.

`setucmp` *cmp* Sets active compartments. Available only to the lowest-level login shell. See `setucmp`(1) for usage and description.

`setulvl` *level* Raises the security level. Available only to the lowest-level login shell. See `setulvl`(1) for usage and description.

`setusrv` Sets the user's security attributes. See `setusrv`(1) for usage and description.

`shift`
`shift` *variable* Shifts the members of *argv* to the left, discarding *argv*. It is an error not to have *argv* set or to have less than 1 word as the value. The second form performs the same function on *variable*.

`source` *name*
`source -h` *name*
Reads commands from *name*. `source` commands can be nested; however, if they are nested too deeply, the shell may run out of file descriptors. An error in a `source` at any level terminates all nested `source` commands. Input during `source` commands is not placed on the history list; the `-h` option causes the commands to be placed in the history list without being executed.

`stop` %*job* ... Stops the specified job that is executing in the background.

`suspend` Causes the shell to stop abruptly, much as if it had been sent a stop signal with `<CONTROL-Z>`. This is most often used to stop shells started by `su`(1).

```
switch (string)
case label:
.
.
.
breaksw
.
.
.
default:
.
.
.
breaksw
endsw
```

Matches each case label successively against *string*, which is the first command and file name expanded. File metacharacters `*`, `?`, and `[...]` may be used in the case labels, which are variable expanded. If none of the labels match before a default label is found, execution would begin after the default label. Each case label and default label must appear at the beginning of a line. The `breaksw` command causes execution to continue after the `endsw;` otherwise, control may fall through case labels and default labels as in C. When no label matches and no `default` exists, execution continues after `endsw.`

`time`

`time` *command* With no argument, the shell prints a summary of time used by this shell and its child processes. When the argument is specified, the shell times the specified simple *command* and prints a time summary as described under the `time` variable. If necessary, an extra shell will be created to print the time statistic when the *command* completes.

`umask`

`umask` *value* The first form of the command displays the file creation mask. The second form sets the file creation mask to the specified value. The mask is specified in octal values. Common values for the mask are 002, which gives all access to the group and read and execute access to others, or 022, which gives all access except write to users in the group and to others.

`unalias` *pattern*

Discards all aliases with names that match *pattern*. All aliases are removed by `unalias *`. The omission of *pattern* is acceptable.

`unhash` Disables the use of the internal hash table. The internal hash table speeds the locating of executed programs.

`unset` *pattern* Removes all variables with names that match *pattern*. To remove all variables, use `unset *`; this has noticeably negative side-effects. The omission of *pattern* is acceptable.

`unsetenv` *pattern*

Removes from the environment all variables with a name that matches *pattern*. See also the preceding `setenv` command and `printenv`(1B).

`wait` Waits for all background jobs. If the shell is interactive, an interrupt can disrupt the wait.

`which` *name* ... Searches for the file that would be executed had *name* been specified as a command. *name* is expanded if it is aliased, and searched for along your path.

`while` *(expr)*
.
.
.
`end`

Although the specified expression evaluates nonzero, the commands between the `while` and the matching `end` are evaluated. Built-in commands `break` and `continue` may be used to terminate or continue the loop prematurely. (`while` and `end` must appear alone on their input lines.) If the input is a terminal, prompting occurs here the first time through the loop as it does with the `foreach` statement.

`%`[*job*][`&`] Brings the current or indicated job to the foreground. When & is included, the job continues to run in the background.

@
@ *name=expr*
@ *name*[*index*]*=expr*

The first form prints the values of all shell variables. The second form sets *name* to the value of *expr*. If the expression contains <, >, &, or |, this part of the expression must be put in parentheses. The third form assigns the value of *expr* to the *index* argument of *name*. Both *name* and its *index* component must already exist.

The operators *=, +=, and so on, are available as in C. You must include either one or more spaces on both sides of the = symbol or no space on either side. Spaces are mandatory in separating components of *expr* that would otherwise be single words.

Special postfix ++ and -- operators increment and decrement *name* respectively (that is, @ i++).

Nonbuilt-in command execution

When a command to be executed is found not to be a built-in command, the shell attempts to execute the command by using execv (see exec(2)). Each word in the PATH variable names a directory from which the shell will attempt to execute the command. When it is not given a -c or -t option, the shell hashes the names in these directories into an internal table so that it executes an exec(2) in a directory only if there is a possibility that the command resides there. This greatly speeds command location when a large number of directories are present in the search path. If this mechanism has been turned off (using unhash), or if the shell was given a -c or -t option (and for each directory component of PATH that does not begin with /), the shell will concatenate with the given command name to form a path name of a file, which it will then attempt to execute.

Parenthesized commands are always executed in a subshell. For example, the following command prints the home directory; leaving you where you were (printing this after the home directory).

```
(cd ; pwd) ; pwd
```

The following command leaves you in the home directory. Parenthesized commands are most often used to prevent chdir from affecting the current shell.

```
cd ; pwd
```

When a path name that has proper execution permissions is found, the shell forks a new process and passes it, along with its arguments to the kernel (using the execv(2) system call). The kernel then attempts to overlay the new process with the desired program. When the file is an executable binary file (see a.out(5)), the kernel succeeds and begins executing the new process. If the file is a text file, and the first line begins with #!, the next word is taken to be the path name of a shell (or command) to interpret that script. Subsequent word(s) on the first line, usually only 1 if any, are used as a single option to that shell. The kernel invokes (overlays) the original shell with the indicated shell, using the name of the script as an argument, which is preceded by the previously mentioned option, if any, and followed by the original user-supplied arguments, again if any. If the indicated shell is a setuid executable file (see setuid(2)), the kernel will not overlay the original shell.

If neither of the preceding conditions holds, the kernel cannot overlay the file (the `execv` call will fail); the C shell then attempts to execute the file by spawning a new shell, as follows:

- If the first character of the file is #, a C shell will be invoked.
- Otherwise, a standard shell (`/bin/sh`) will be invoked.

When there is an alias for *shell*, the words of `alias` are prepended to the argument list to form the shell command. The first word of `alias` should be the full path name of the shell (for example, `$shell`). This is a special, late-occurring case of `alias` substitution, and it allows only words to be prepended to the argument list without modification.

Jobs

The shell associates a job with each pipeline. It keeps a table of current jobs, printed by the `jobs` command, and assigns them small integer numbers. When a job is started asynchronously with &, the shell prints the following line:

```
[1] 1234
```

This line indicates that the job that was started asynchronously, was job number 1, and it had one (top-level) process, with a process ID of 1234.

The shell redirects standard input to `/dev/null` for a job being run in the background. This results in the receipt of an end-of-file if such a job tries to read from the terminal. Background jobs are usually allowed to produce output.

There are several ways to refer to jobs in the shell. The character % introduces a job name. If you wish to refer to job number 1, you can name it `%1`. Jobs can also be named by prefixes of the string entered in to start them if these prefixes are unambiguous. It is also possible to enter %?*string*, which specifies a job with text that contains *string* if there is only one such job.

The shell maintains the current and previous jobs. In output pertaining to jobs, the current job is marked with a + symbol and the previous job with a - symbol. The abbreviation `%+` refers to the current job, and `%-` refers to the previous job. The characters %% are a synonym for the current job.

Status Reporting

The shell learns immediately whenever a process changes state. It usually informs you, just before it prints a prompt, whenever a job becomes blocked so that no further progress is possible. This is done so that it does not otherwise disturb your work. If you set shell variable `notify`, the shell notifies you immediately of changes of status in background jobs. There is also a shell command called `notify`, which marks a single process so that its status changes are immediately reported. By default, `notify` marks the current process; simply enter `notify` after starting a background job to mark it.

Substitutions

The following subsections describe the various transformations the shell performs on input. The shell performs these in the following order:

- History substitution
- Alias substitution

- Variable substitution
- Command substitution
- File-name substitution

History substitution

History substitutions place words from previous command input as portions of new commands, making it easy to repeat commands, repeat arguments of a previous command in the current command, and correct spelling mistakes in the previous command with little typing and a high degree of confidence. History substitutions begin with `!` and may begin anywhere in the input stream (as long as they do not nest.) The `!` can be preceded by `\` to override its special meaning; for convenience, `!` is passed unchanged when it is followed by a blank, tab, new-line character, `=`, or `(`. Note that `!~` is a `csh` operator and cannot be used for history substitution. (History substitutions also occur when an input line begins with a carat (`^`), described later.) Any input line that contains history substitution is echoed on the terminal before it is executed because it could have been typed without history substitution.

Commands input from the terminal consisting of 1 or more words are saved in the history list. The history substitutions reintroduce sequences of words from these saved commands into the input stream (the size of which is controlled by the `history` variable). The previous command is always retained, regardless of its value. Commands are numbered sequentially from 1.

Consider the following output from the `history` command:

```
 9  write michael
10  ex write.c
11  cat oldwrite.c
12  diff *write.c
```

The commands are shown with their event numbers. It is not usually necessary to use event numbers, but you can make the current event number part of the prompt by placing `!` in the prompt string.

With the current event 13, you can refer to previous events by event number, such as `!11`; relatively, as in `!-2` (referring to the same event); by a prefix of a command word, as in `!d` for event 12 or `!wri` for event 9; or by a string contained in a word in the command as in `!?mic?`, which also refers to event 9. These forms, without further modification, reintroduce the words of the specified events, each separated by a blank character. As a special case, `!!` refers to the previous command; thus `!!` alone is essentially `redo.`

To select words from an event, add a colon (`:`) and a designator for the desired words to the event specification. The words of an input line are numbered starting at 0. The first (usually command) word is 0, the second (first argument) is 1, and so on. The basic word designators are as follows:

`0` First (command) word

n *n*th argument

`^` First argument, that is, `1`

`$` Last argument

`%` Word matched by (immediately preceding) ?*s*? search

x–*y*	Range of words
–*y*	Abbreviation `0-`*y*
*	Abbreviation `^-$`, or nothing if only 1 word in event
x *	Abbreviation *x*–`$`
x–	Like *x** but omits word `$`

The colon separating the event specification from the word designator can be omitted if the argument selector begins with `^`, `$`, `*`, `-`, or `%`. After the optional word designator, you can place a sequence of modifiers, each preceded by a colon. The following modifiers are defined:

`h`	Removes trailing path name component, leaving the head
`r`	Removes trailing *.xxx* component, leaving the root name
`e`	Removes all but the extension *.xxx*
`s/`*l*`/`*r*`/`	Substitutes *r* for *l*
`t`	Removes all leading path name components, leaving the tail
`&`	Repeats the previous substitution
`g`	Applies the change globally, prefixing & (for example, `g&`)
`p`	Prints the new command but does not execute it
`q`	Places quotation marks around the substituted words, preventing further substitutions
`x`	Like `q`, but breaks into words at blanks, tabs, and new-line characters

Unless preceded by `g`, the modification is applied to only the first modifiable word. With substitutions, it is an error for no word to be applicable.

The left-hand side of substitutions are strings rather than regular expressions in the sense of the editors. Any character may be used as the delimiter in place of the `/` symbol; a `\` places quotation marks around the delimiter in the *l* and *r* strings. The character & in the right-hand side is replaced by the text from the left. A `\` quotes & also. A null *l*" uses the previous string, either from a *l*" or from a contextual scan string *s*" in `!?`*s*`?`. The trailing delimiter in the substitution may be omitted if a new line follows immediately, as may the trailing `?` in a contextual scan.

A history reference may be given without an event specification (for example, `!$`). In this case, the reference is to the previous command unless a previous history reference occurred on the same line, in which case this form repeats the previous reference. For example,`!?foo?^ !$` gives the first and last arguments from the command matching `?foo?`.

A special abbreviation of a history reference occurs when the first nonblank character of an input line is a caret (^). This is equivalent to `!:s^`, providing a convenient shorthand for substitutions in the text of the previous line. For example, `^lb^lib` fixes the spelling of `lib` in the previous command. Finally, a history substitution may be surrounded with braces `{}` if necessary to insulate it from the characters that follow. For example, after `ls -ld ~paul`, if you use `!{l}a` to do `ls -ld ~paula`, `!la` will look for a command starting with `la`.

Quotations with ′ and "

You can put single and double quotation marks around strings (′ and ") to override all or some of the remaining substitutions. Strings enclosed in ′ are prevented from any further interpretation. Strings enclosed in " may be expanded as described in the following.

In both cases, the resulting text becomes (all or part of) a single word; only in one special case (see the Command Substitution subsection) does a string enclosed in " yield parts of more than 1 word. Strings enclosed in ′ never yield parts of more than 1 word.

Alias substitution

The shell maintains a list of aliases that can be established, displayed, and modified by the `alias` and `unalias` commands. After a command line is scanned, it is parsed into distinct commands, and the first word of each command is checked left-to-right to see whether it has an alias. If it does, the text that is the alias for that command will be reread, with the history mechanism available as though that command were the previous input line. The resulting words replace the command and argument list. If no reference is made to the history list, the argument list will remain unchanged.

For example, if the alias for `ls` were `ls -l`, the command `ls /usr` would map to `ls -l /usr`; the argument list would be undisturbed. Similarly, if the alias for `lookup` were `grep !^ /etc/passwd`, `lookup bill` would map to `grep bill /etc/passwd`.

When an alias is found, the word transformation of the input text is performed and the aliasing process repeats on the new input line. If the first word of the new text is the same as the old, looping is prevented by flagging it to prevent further aliasing. Other loops are detected and cause an error.

The mechanism allows aliases to introduce parser metasyntax. Thus, you can use `alias print 'pr \!* | exlp'` to make a command that `pr`'s its arguments to the line printer.

Variable substitution

The shell maintains a set of variables, each of which has as its value a list of zero or more words. Some of these variables are set by the shell or referred to by it. For instance, the *argv* variable is an image of the shell's argument list, and words of this variable's value are referred to in special ways.

The values of variables may be displayed and changed by using the `set` and `unset` commands. Of the variables referred to by the shell, several are toggling; the shell disregards their value, checking only to see whether they are set. For instance, toggling the `verbose` variable causes command input to be echoed. The setting of this variable results from the `-v` command-line option.

Other operations treat variables numerically. The @ command permits numeric calculations to be performed and the result assigned to a variable. Variable values are, however, always represented as zero or more strings. For the purposes of numeric operations, the null string is considered to be zero, and the second and subsequent words of multiword values are ignored.

After the input line is aliased and parsed, and before each command is executed, variable substitution is performed by the $ character. You can prevent this expansion by preceding the $ with a \ except within double quotation marks (") where it always occurs, and within single quotation marks (') where it never occurs. Strings enclosed by ` are interpreted later (see the Command Substitution subsection); $ substitution does not occur there until later, if at all. $ is passed unchanged if it is followed by a blank, tab, or end-of-line character.

Input/output redirections are recognized before variable expansion, and they are variable expanded separately. Otherwise, the command name and entire argument list are expanded together. It is possible for the first command word to this point to generate more than 1 word, the first of which becomes the command name, and the rest of which become arguments.

Unless enclosed in " or given the :q modifier, the results of variable substitution may eventually be command and file-name-substituted. A variable within ", whose value consists of multiple words, expands to a portion of a single word, with the words of the variable's value separated by blanks. When the :q modifier is applied to a substitution, the variable expands to multiple words, with each word separated by a blank and enclosed in quotation marks to prevent later command or file name substitution.

The following metasequences introduce variable values into the shell input. Except as noted, it is an error to reference a variable that is not set.

$*name*
${*name*} Replaced by the words of the value of variable *name*, each separated from the others by a blank. Braces insulate *name* from following characters, which would otherwise be part of it. Shell variables have names consisting of up to 18 letters and digits starting with a letter. The underscore character is considered a letter. If *name* is not a shell variable, but is set in the environment, that value will be returned (but : modifiers and the other forms specified are not available in this case).

$*name*[*selector*]
${*name*[*selector*]}
May be used to select only some of the words from the value of *name*. The selector is subjected to $ substitution and may consist of either a single number or two numbers separated by a - symbol. The first word of a variable's value is numbered 1. If the first number of a range is omitted, it defaults to 1. If the last member of a range is omitted, it defaults to $#*name*. The selector * selects all words. It is legal for a range to be empty if the second argument is omitted or in range.

$#*name*
${#*name*} Specifies the number of words in the variable. This is useful later in [*selector*].

$0 Substitutes the name of the file from which command input is being read. An error occurs if the name is unknown.

$*number*
${*number*} Equivalent to `$argv[`*number*`]`.

$* Equivalent to `$argv[*]`.

Modifiers `:h`, `:t`, `:r`, `:q`, and `:x` may be applied to the preceding substitutions (except for `$0`) as may `:gh`, `:gt`, and `:gr`. If braces `{ }` appear in the command form, the modifiers must appear within the braces. The current implementation allows only one `:` modifier on each `$` expansion.

The following substitutions may not be modified with `:` modifiers:

$?*name*
${?*name*} Substitutes 1 if name is set, 0 if it is not.

$?0 Substitutes 1 if the current input file name is known, 0 if it is not.

$$ Substitutes the (decimal) process number of the (parent) shell.

$< Substitutes a line from standard input, with no further interpretation thereafter. It can be used to read from the keyboard in a shell script.

Command and file-name substitution

Command and file-name substitution are applied selectively to the arguments of built-in commands. That is, portions of expressions that are not evaluated are not subjected to these expansions. For commands that are not internal to the shell, the command name is substituted separately from the argument list. This occurs in a child of the main shell after input-output redirection is performed.

Command substitution

Command substitution is indicated by a command enclosed in single quotation marks (`). The output is broken into separate words at blanks, tabs, and new lines, and null words are discarded. This text then replaces the original string. Within double quotation marks ("), only new lines force new words; blanks and tabs are preserved.

In any case, the single final new line does not force a new word. Note that this makes it possible for a command substitution to yield only part of a word, even if the command outputs a complete line.

File-name substitution

If a word contains any of the characters `*`, `?`, `[`, or `{`, or begins with the character `~`, that word is a candidate for file-name substitution (also known as *globbing*). This word is then regarded as a pattern and is replaced with an alphabetically sorted list of file names that match the pattern. In a list of words specifying file name substitution, it is an error if no pattern matches an existing file name, but it is not required that each pattern match. Metacharacters `*`, `?`, and `[` are the only ones that imply pattern matching; the characters `~` and `{` are more akin to abbreviations.

In matching file names, the dot character (`.`) at the beginning of a file name or immediately following a `/`, as well as the character `/`, must be matched explicitly. The character `*` matches any string of characters, including the null string. The character `?` matches any single character. The sequence `[...]` matches any one of the characters enclosed. Within `[...]`, a pair of characters separated by `-`, matches any character lexically between the two.

The ~ character at the beginning of a file name is used to refer to home directories. When the ~ is alone, it expands to the user's home directory as reflected in the value of the variable `home`. When followed by a name consisting of letters, digits, and - characters, the shell searches for a user with that name and substitutes that user's home directory; for example, `~ken` might expand to `/usr/ken` and `~ken/chmach` to `/usr/ken/chmach`. If the ~ character is followed by a character other than a letter or `/`, or does not appear at the beginning of a word, it is left undisturbed.

Metanotation `a{b,c,d}e` is shorthand for `abe ace ade`. Left-to-right order is preserved, with results of matches being sorted separately at a low level to preserve this order. This construction may be nested. For example, `~source/s1/{oldls,ls}.c` expands to `/usr/source/s1/oldls.c /usr/source/s1/ls.c`, whether or not these files exist, without any chance of error if the home directory for `source` is `/usr/source`. Similarly `../{memo,*box}` might expand to `../memo ../box ../mbox`. (Note that `memo` was not sorted with the results of matching `*box`.) As a special case, `{`, `}`, and `{}` are passed undisturbed.

Input/Output

The standard input and output of a command may be redirected with the following syntax:

`< name` — Opens file *name* (which is first variable, command, and file-name expanded) as the standard input.

`<< word` — Reads the shell input up to a line that is identical to *word*. *word* is not subjected to variable, file-name, or command substitution, and each input line is compared to *word* before any substitutions are done on this input line. Unless a quoting \, ", ´, or ` appears in *word*, variable and command substitution are performed on the intervening lines, allowing \ to add quotation marks to $, \, and `. Commands that are substituted have all blanks, tabs, and new lines preserved, except for the final new line, which is dropped. The resultant text is placed in an anonymous temporary file that is given to the command as standard input.

`> name`
`>! name`
`>& name`
`>&! name` — The file *name* is used as standard output. If the file does not exist, it will be created; if the file exists, it will be truncated, losing its previous contents.

If the variable `noclobber` is set, the file must not exist or be a character special file (for example, a terminal or `/dev/null`); if it is, an error will result. This helps prevent accidental destruction of files. In this case, the `!` forms can be used to suppress this check.

The forms involving `&` route the diagnostic output into the specified file as well as the standard output. *name* is expanded in the same way as `<` input file names are expanded.

`>> name`
`>>& name`
`>>! name`
`>>&! name` — Uses file *name* as standard output, for example, `>`, but places output at the end of the file. If the variable `noclobber` is set, the file must not exist unless one of the `!` forms is specified; otherwise, it is similar to `>`.

A command receives the environment in which the shell was invoked, as modified by the input-output parameters and the presence of the command in a pipeline. Unlike the case in some previous shells, commands run from a file of shell commands have no access to the text of the commands by default; instead they receive the original standard input of the shell. The `<<` mechanism should be used to present in-line data. This permits shell command scripts to function as components of pipelines and allows the shell to block read its input. The default standard input for a command run detached remains as the original standard input of the shell. If this is a terminal and if the process attempts to read from the terminal, the process will block and the user will be notified.

Diagnostic output may be directed through a pipe with the standard output by using the form `|&` rather than `|`.

Expressions

A number of the built-in commands take expressions, in which the operators are similar to those of C, with the same precedence. Within an individual precedence group, however, the expressions are evaluated from right to left, as opposed to the standard left to right. These expressions appear in the `@`, `exit`, `if`, and `while` commands. The following operators are available:

```
||  &&  |  ^ &  ==  !=  =~  !~  <=  >=  <  >  <<  >>  +  -  *  /  %  !  ~  (  )
```

The precedence increases to the right, `==`, `!=`, `=~`, and `!~`, `<=`, `>=`, `<`, and `>`, `<<`, and `>>`, `+`, and `-`, `*`, `/`, and `%` being, in groups, at the same level. The `==`, `!=`, `=~`, and `!~` operators compare their arguments as strings; all others operate on numbers. The operators `=~` and `!~` are like `!=` and `==` except that the right-hand side is a pattern (containing, for example, `*`, `?`, and instances of `[...]`) against which the left-hand operand is matched. This reduces the need for the *switch* statement in shell scripts when all that is really needed is pattern matching.

Strings that begin with 0 are octal numbers; valid octal digits are 0 through 7. Strings that begin with `0x` or `0X` are hexadecimal numbers; valid hexadecimal digits are 0 through 9, a through f, and A through F. Null or missing arguments are considered 0. The results of all expressions are strings, which represent decimal numbers.

No two components of an expression can appear in the same word; except when adjacent to components of expressions that are syntactically significant to the (`& | < > ( )`) parser, they are surrounded by spaces.

Also available in expressions as primitive operands are command executions enclosed in braces `{}` and file inquiries of the form `-` *lname*;

l is one of the following:

`r`	Read access
`w`	Write access
`x`	Execute access
`e`	Existence
`l`	Symbolic link
`o`	Ownership
`z`	Zero size
`f`	Plain file

m Migrated file (type IFOFL)
M Migrated file (has a DMF handle)
d Directory

The specified name is command and file name expanded and then tested to see whether it has the specified relationship to the real user. If the file does not exist or is inaccessible, all inquiries return false (that is, 0). Successful command executions return the value 1 if true or 0 if false. If more detailed status information is required, the command should be executed outside an expression and the variable `status` examined.

Control Flow

The shell contains a number of commands that can be used to regulate the flow of control in command files (shell scripts) and, in limited but useful ways, from terminal input. These commands all operate by forcing the shell to reread or skip in its input and, because of the implementation, restrict the placement of some of the commands.

The `foreach`, `switch`, and `while` statements, as well as the `if-then-else` form of the `if` statement require that the major keywords appear in a single simple command on an input line.

If the shell's input is to be searched, the shell buffers input whenever a loop is being read and performs seeks in this internal buffer to accomplish the rereading implied by the loop. (To the extent allowed, backward `goto` statements succeed on nonseekable inputs.)

Predefined and Environment Variables

Some variables have special meaning to the shell. Of these, the shell always sets `argv`, `cwd`, `home`, `path`, `shell`, and `status`. The `prompt` variable is always set in an interactive shell. Except for `cwd` and `status`, these variables are set only at initialization; therefore, if you wish to modify these, you must do so explicitly.

The shell copies the environment variable `LOGNAME` into the variable `user`, `TERM` into `term`, and `HOME` into `home;` these are then copied back into the environment whenever the normal shell variables are reset. The environment variable `PATH` is handled likewise; you do not have to worry about its setting other than in the `.cshrc` file because child `csh` processes reset the path variable when you use the `source` command on the `.cshrc` file.

`argv` Sets the arguments to the shell. Positional parameters are substituted from this variable; for example, `$1` is replaced by `$argv[1]`.

`cdpath` Provides a list of alternative directories that are searched to find subdirectories in `chdir` commands.

`cwd` Provides the full path name of the current directory.

`echo` Set when the `-x` command-line option is specified. Causes each command and its arguments to be echoed just before execution. For nonbuilt-in commands, all expansions occur before echoing. Built-in commands are echoed before command and file-name substitution because these substitutions are then done selectively.

`histchars` Can be given a string value to change the characters used in history substitution. The first character of its value is used as the history substitution character, replacing default character `!`. The second character of its value replaces the character `^` in quick substitutions.

`history` Can be given a numeric value to control the size of the history list. Any command that has been referenced in this number of events is not discarded. If the value of `history` is too large, the shell may run out of memory. The last executed command is saved on the history list.

`home` The user's home directory, initialized from the environment. The file-name expansion of `~` refers to this variable. Setting `home` to a path name that begins with `.` or `..` causes errors.

`ignoreeof` When set, the shell ignores end-of-file from input devices, which are terminals. This prevents shells from accidentally being killed by one or more `<CONTROL-d>`.

`mail` Specifies the files in which the shell checks for mail. This is done after each command completion, which results in a prompt if a specified interval has elapsed. The shell displays `You have new mail` if the file exists with an access time not greater than its modification time.

When the first word of the value of `mail` is numeric, it specifies a different mail checking interval, in seconds, instead of the default, which is 10 minutes.

When multiple mail files are specified, the shell displays `New mail in` *name* when there is mail in file *name*.

`noclobber` As described in the section on input/output, restrictions are placed on output redirection to ensure that files are not accidentally destroyed, and that `>>` redirections refer to existing files.

`noglob` When set, inhibits file-name expansion. This is most useful in shell scripts that are not dealing with file names, or when a list of file names has been obtained and further expansions are not desirable.

`nonomatch` When set, specifies that a file-name expansion does not have to match any existing files; rather the primitive pattern is returned. It is still an error, however, for the primitive pattern to be malformed; that is, `echo [` still produces an error.

`notify` When set, the shell notifies the user of job completions asynchronously. The default is to present job completions just before printing a prompt.

`path` Each word of the path variable specifies a directory in which commands are to be sought for execution. A null word specifies the current directory. If there is no `path` variable, only full path names will execute. The usual search path is `/bin`, `/usr/bin`, and `/usr/ucb`, but this may vary from system to system. For the super user, the default search path is `/etc`, `/bin`, and `/usr/bin`. A shell that is given neither the `-c` nor the `-t` option usually hashes the contents of the directories in the `path` variable after reading `.cshrc` and also each time the `path` variable is reset. If new commands are added to these directories while the shell is active, it may be necessary to specify the `rehash` command; otherwise, the commands may not be found.

`prompt` The string printed before each command is read from an interactive terminal input. If `!` appears in the string, it will be replaced by the current event number unless a preceding `\` is given. The default is % or # for the super user.

`savehist` Contains a numeric value that controls the number of entries of the history list that are saved in `~/.history` when the user logs out. Any command referenced in this number of events is saved. During startup, the shell sources `~/.history` into the history list, enabling history to be saved across logins. Values of `savehist` that are too large can slow the shell during startup.

`shell` The file in which the shell resides. This is used in forking shells to interpret files that have execute bits set that are not executable by the system. (See the description of nonbuilt-in command execution.) This file is initialized to the (system-dependent) home of the shell.

`status` The status returned by the last command. If it terminated abnormally, 0200 is added to the status. Built-in commands that fail return exit status 1; all other built-in commands set the status to 0.

`time` Controls the automatic timing of commands. If it is set, any command that takes more than this number of CPU seconds prints a line specifying user, system, and real times. It also prints a usage percentage, which is the ratio of user plus system times to real time that is printed when it terminates.

`timestamp` Set by the `-S` command-line option. Causes a date and time stamp in the form *day month date hh:mm:ss* to be `echo`ed before each command is executed.

`verbose` Set by the `-v` command-line option, which prints the words of each command after history substitution.

Nonbuilt-in Command Execution

When a command to be executed is found to not be a built-in command, the shell attempts to execute the command through `execv` (see `exec`(2)). Each word in the variable *path* names a directory from which the shell attempts to execute the command. If neither option `-c` or option `-t` is specified, the shell will hash the names in these directories into an internal table so that it will try an `exec` in a directory only if there is a possibility that the command resides there. This greatly speeds command location when a large number of directories are present in the search path. If this mechanism has been turned off (by `unhash`), or if `-c` or `-t` was specified, for each directory component of *path* that does not begin with a `/`, the shell will concatenate with the specified command name to form a path name of a file, which it will then attempt to execute.

Signal Handling

The shell ignores `quit` signals. Jobs running detached (either by `&` or the `bg` or `%. . . &` commands) are immune to signals generated from the keyboard, including hangups. Other signals have the values that the shell inherited from its parent. `onintr` can control the shell's handling of interrupts and terminate signals in shell scripts. Login shells catch the `terminate` signal; otherwise, this signal is passed on to children from the state in the shell's parent. Interrupts are not allowed when a login shell is reading the `.logout` file.

NOTES

Words can be no longer than 1024 characters. The system limits argument lists to 50,000 characters. The number of arguments to a command involving file-name expansion is limited to one-sixth the number of characters allowed in an argument list. Command substitutions cannot substitute more characters than are allowed in an argument list.

The `csh` utility does not send messages about illegal options.

The `csh` metacharacters will not match those characters that are greater than or equal to 0200.

The C shell will not execute the last command line in a script file if the last line is not terminated with a newline character. Some editors, such as `emacs`(1), are capable of creating such files.

BUGS

When a command is restarted from a stop, the shell prints the directory in which it started if this is different from the current directory; this can be incorrect because the job may have changed directories internally.

Shell built-in functions cannot be stopped and restarted. Command sequences of the form `a ; b ; c` are also not handled gracefully when stopping is attempted. If you suspend `b`, the shell will then immediately execute `c`. This is especially noticeable if this expansion results from an alias. It suffices to place the sequence of commands in parentheses to force it to a subshell, that is, `( a ; b ; c )`.

Control over tty output after processes are started is primitive.

Alias substitution is most often used to simulate shell scripts, but this is not efficient; shell scripts should be provided.

Commands within loops, prompted by ?, are not placed in the history list. Control structure should be parsed rather than being recognized as built-in commands. This would allow control commands to be placed anywhere, to be combined with |, and to be used with & and ; metasyntax.

It may be possible to use the : modifiers on the output of command substitutions. All and more than one : modifier may be allowed on $ substitutions.

Although set and setenv allow the definition of variable names that do not begin with a letter or underscore or that are longer than 18 characters, use of such variables results in an error.

If onintr follows onintr - in a shell script, interrupts continue to be ignored.

A job started asynchronously with & and containing command substitution is not protected from interrupts.

FILES

~/.cshrc	Read at beginning of execution by each shell
~/.login	Read by login shell after .cshrc at login
~/.logout	Read by login shell at logout
/bin/sh	Standard shell
/etc/passwd	Source of home directories for ~*name*
/etc/cshrc	Read at beginning of execution of shell, before .cshrc
/tmp/sh*	Temporary file for <<

(See the File-name Substitution subsection for a discussion of the ~ character in file names.)

SEE ALSO

setucat(1), setucmp(1), setulvl(1), setusrv(1), sh(1)

access(2), exec(2), fork(2), kill(2), pipe(2), setuid(2), signal(2), umask(2), wait(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

a.out(5), cshrc(5) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

dmmode(1) Online only

NAME

csort – Sorts and/or merges blocked files

SYNOPSIS

csort [-e] [-k *keyfile*] [-l *statfile*] [-r] *outfile sortfiles*

csort [-e] [-k *keyfile*] [-l *statfile*] -m *mfiles* [-n] [-r] *outfile* [*sortfiles*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The csort utility orders records of unsorted *sortfiles* and/or previously sorted -m *mfiles* and writes the result to the file named by the required *outfile* specification. The csort utility reads and writes files, depending on the sort keys you provide. If every key field is either a DEC key or a CHR key, input and output files are considered to be character files. If any key is BIN, FLT, or INT, the input and output files are considered to be Fortran unformatted files.

Two synopses for csort are provided to show that csort can be invoked only with either unsorted *sortfiles*, or with previously sorted *mfiles* and optionally one or more *sortfiles*. Invocation of csort with only the *outfile* operand will be unsuccessful.

The csort utility accepts the following options:

-e Writes the directives from *keyfile* into *statfile*.

-k *keyfile* The *keyfile* argument names the file that contains the directives to be used to determine tuning parameters and/or keys to determine the sorting scheme. For information about these directives, see the DIRECTIVES section.

If the -k option is not specified, csort provides default values for the sorting process. These defaults are as follows:

- Key type is a character string.
- Ascending sort order.
- Key field starts with the first character of the first 64-bit word of the record.
- Key field ends with the last character of the first 64-bit word of the record.
- ASCII collating sequence.

When there are multiple sort keys, later keys are compared only if earlier keys are equal.

-l *statfile* The *statfile* argument names the file to contain the listing of the statistics on the sorting process. The default file name is SRTSTAT.

`-m` *mfiles* The files specified by *mfiles* are assumed to be sorted and ready for merging. They will be merged with each other and with intermediate output from the sorting process of the unsorted *sortfiles*.

The files named in *mfiles* can be designated by using one of the following forms:

- List of files separated from one another by a comma.
- List of files enclosed in quotation marks and separated by commas and/or white space.

`-n` The files specified by *mfiles* are not checked to verify whether they have been sorted previously. The `-m` option must also be specified.

`-r` Retains the original input order of a sequence of records with equivalent keys.

outfile Specifies the file where the results of `csort` are stored.

sortfiles Specifies the files to sort and/or merge.

For suitably large files, `csort` will require temporary files. By default, these files are created in the directory specified by your `TMPDIR` environment variable. In all but the most demanding applications, this will work well.

In some cases, however, you may need a greater degree of control over the allocation of temporary files. You may initialize the environment variable `CSORTDIR` to specify the directories within which temporary files are created.

`CSORTDIR` is a colon-separated list of directory names. As temporary files are needed, they are created round-robin within the provided list of directories. (If `CSORTDIR` is defined but null, temporary files will be created in your current directory, which may or may not be desirable.

DIRECTIVES

The two `csort` directives are `KEY` and `TUNE`. The `KEY` directive lets you define the keys that determine the major and minor fields on which a sort is to be performed. The `TUNE` directive permits you to optimize the sort operation. Only `KEY` is required. The directives must be in a file specified with the `-k` option on the `csort` command line.

KEY Directive

The required `KEY` directive defines the keys of the major and minor field on which a sort is to be performed. The major field is the one on which `csort` first orders the records. If two or more records contain identical major fields, `csort` uses the minor fields to order them.

The `KEY` directive can appear any time and as often as necessary. The number of specifications is unlimited. Keys are given priority by the order in which they are specified.

The format of the `KEY` directive is as follows:

`KEY,TYPE=`*type*`,ORDER=`*order*`,START=`*w*`:`*c*`:`*b*`,END=`*w*`:`*c*`:`*b*`,COLSEQ=`*colseq*

`TYPE=`*type* Specifies the form in which the key appears in the record; *type* can be one of the following:

`BIN` Binary bit stream (unsigned).

`FLT` Floating-pont number (64-bit Cray internal format).

`INT` Integer value (a maximum of 64 bits in twos complement Cray format).

`CHR` Character string. The string's field is the length defined by the `START` and `END` parameters. The default character size is 8 bits; word size is 64 bits. This key causes `csort` to read the file in character mode, decompressing blanks.

`DEC` Decimal number (real number with a decimal point or integer) in ASCII or EBCDIC characters. `csort` does not accept real numbers with exponents. Leading and trailing blanks are insignificant; embedded blanks are interpreted as 0. The + or – sign precedes the number or decimal point. This key causes `csort` to read the file in character mode, decompressing blanks.

If you specify `DEC`, `csort` converts all numbers to floating-point numbers. Thus, `10` and `10.` can be used interchangeably.

`ORDER=`*order* Specifies the order in which `csort` sorts the key; *order* can be either `ASCEND` (sorts the records in ascending order by the key rank; default) or `DESCEND` (sorts the records in descending order by the key rank).

`START=`*w*`:`*c*`:`*b* Required parameter; no defaults. The starting position of the field containing the key. The end position is not required for floating-point or integer keys.

If you specified `FLT` or `INT` for the key *type*, the key defaults to 64 bits; therefore, you do not need to specify an end position unless the key length is other than 64 bits. The starting position specifiers are the only key location identifiers necessary.

Specify integer numbers as follows:

w Starting position of a word

c Starting position of a character

b Starting position of a bit

Count words, characters, and bits from the left beginning with 1, not 0. You can define a field as a character position within a word (*w:c*). The following example defines the seventh character of the third word:

```
START=3:7
```

You can also define a field as a character position without a word position by putting an asterisk in place of the omitted unit. The following example is equivalent to the previous example:

```
START=*:23
```

Because each word is 8 characters, the seventh character of the third word is the twenty-third character.

Similarly, you can specify a bit position directly in a word or in a character position.

END=*w*:*c*:*b* Indicates the ending position of the field containing the key. The END parameter is required only with sequence types BIN, CHR, and DEC. END is optional with FLT and INT. If the type of the key is FLT or INT, the default is the beginning of the field as specified by the START directive plus 64 bits.

Specify integer numbers as follows:

w Ending position of a word

c Ending position of a character

b Ending position of a bit

COLSEQ=*colseq* Specifies the collating sequence for the CHR or DEC key type. The choices are the following:

ASCII Sorts according to the ASCII sequence (default).

ASCIIUP Sorts according to the ASCII sequence except that lowercase letters are treated as if they are uppercase.

EBCDIC Sorts ASCII characters according to the EBCDIC sequence.

EBCDICUP Sorts ASCII characters according to the EBCDIC sequence except that lowercase letters are treated as if they are uppercase.

EBCDIC and EBCDICUP are used when the ASCII characters in the output file are to be translated from ASCII to EBCDIC (for example, during transfer to a front-end machine that used EBCDIC). The resulting translated file then has EBCDIC characters in EBCDIC sequence. If the Cray file contains EBCDIC characters that have not been converted to ASCII, the ASCII or ASCIIUP sequences (which order elements based on the numerical bit value) yield the expected results.

The ASCII, ASCIIUP, EBCDIC, and EBCDICUP collating sequences are the only ones available with the csort control statement. To define any other collating sequence, you must use the csort subroutines and identify the collating sequence with a SAMSEQ call.

TUNE Directive

The optional TUNE directives optimizes csort through the efficient allocation of resources. To judge the effect of the options, you need the statistics found in the dataset indicated by the L parameter on the csort control statement. The TUNE directive can appear as often as necessary. If a parameter is speciefed more than once in different directives, the last value specified takes precedence over the previous ones.

Use the TUNE directive to specify information about the files to be sorted and resources to be allocated. When you make these specifications, accuracy is important; inaccurate estimates can degrade performance.

The format of the TUNE directive is as follows:

```
TUNE,AVRL=n,MXRL=n,NRECEST=n, DISK=name:name...,DSLO=n,
NAMEBM=namebm,NAMESSD=namessd,NDS=n,
NDSSD=n,NBSSD=n,NDBM=n,NBBM=n,NBDSK=n, MNBL=n,MXBL=n
```

AVRL=*n*	Specifies the average record length. The default is the maximum record length.
MXRL=*n*	Specifies the maximum record length. The default is 20 Cray words (160 bytes). If the maximum record length is too short, csort aborts during the input phase.
NRECEST=*n*	An estimate of the number of records in the input files. The default is 1,000,000 records. This value is no longer used.
DISK=*name:name...*	No longer used in this implementation. Instead, you may specify a colon-separated list of directories in the CSORTDIR environment variable.
DSLO=*n*	No longer used in this implementation.
NAMEBM=*namebm*	No longer used in this implementation.
NAMESSD=*namessd*	No longer used in this implementation.
NDS=*n*	Specifies the number of temporary datasets to be used during the merge phase. The default is 10; minimum is 4. Change this parameter only if you must run Sort/Merge in minimum memory. A large value is recommended, but many temporary datasets are assigned smaller buffers and buffers should be large to maximize I/O efficiency. This makes it difficult to assign an efficient number.
NDSSD=*n*	No longer used in this implementation.
NBSSD=*n*	No longer used in this implementation.
NDBM=*n*	No longer used in this implementation.
NBBM=*n*	No longer used in this implementation.
NBDSK=*n*	Specifies the number of sort buffers to be allocated to each temporary dataset. The size of the buffer is specified by the MNBL= and MXBL= parameters. The default value is 2.
MNBL=*n*	Specifies the number of word blocks in each sort buffer. The default is 42 (the track size of a DD-49). If you supply both a minimum and a maximum, the minimum must be the smaller of the two numbers.
MXBL=*n*	Specifies the number of word blocks in each sort buffer. The default is 42. If you supply both a minimum and a maximum, the minimum must be the smaller of the two numbers.

NOTES

When calling any of the sort routines from within an applications program, you must specify that the sort library, `/usr/lib/libsort`, be loaded along with your program and the standard system default libraries. To do this, you can use the following command line (provided that your program requires no other special loading options):

`segldr -lsort` *your_program*

EXAMPLES

Example 1: This example sorts the contents of `infile1` and `infile2`, placing the output in `outfile` and using the default sorting key:

```
csort outfile infile1 infile2
```

Example 2: This example sorts, in reverse order, the records found in `infile1` and `infile2`, placing the output in `outfile` and using only the first character of the second word as the sort key.

```
csort -k keyfile outfile infile1 infile2
```

The key information is obtained from this keyfile:

```
key,type=chr,order=descend,start=2:1,end=2:2.
```

Example 3: This example merges the records of the already sorted files `sorted1` and `sorted2` along with the sorted output from the unsorted file `unsort1`, placing the output in `outfile` and using the integer located in the first possible position of the record as the sort key.

```
$ csort -m sorted1,sorted2 -k keyfile outfile unsort1
```

The key information is obtained from this keyfile:

```
key,type=int,start=1:1.
```

Example 4: This example sorts direct access files:

```
#!/bin/csh
rm -rf mkdata.f srtfile outfile keyfile
cat > mkdata.f << EOF
      program makedata
      implicit integer (A-Z)
      character*8 junk
30    FORMAT(I5)
      write(junk,'(A8)') 'srtfile'L
      open ( unit=8, file=junk , access='direct',
     C       form='unformatted', status='NEW' , recl=24 )
      iseed = IRTC()
      CALL RANSET(iseed)
      DO 10 I=1,10
```

```
      K=INT(RANF()*10000+10000)
      x=i*4.0
      y=x/2.0
      WRITE(8,REC=I) K, X, Y
10    continue
      close(8)
      stop
      end
EOF
#
cat > keyfile << EOF
key,type=int,start=1:1:1.
EOF
#
cf77 -o mkdata mkdata.f
#
./mkdata
#
assign -F ibm.f:24 srtfile
assign -F ibm.f:24 outfile
#
csort -k keyfile outfile srtfile
#
echo "============================================================"
echo "                    UNSORTED INPUT"
echo "============================================================"
od srtfile
#
echo "============================================================"
echo "                    SORTED INPUT"
echo "============================================================"
od outfile
```

FILES

SRTSTAT Default *statfile*

SEE ALSO

sort(1)

SAMKEY(3F), SAMTUNE(3F) in the *UNICOS System Libraries Reference Manual*, Cray Research publication SR–2080

NAME

csplit – Splits files based on context

SYNOPSIS

csplit [-f *prefix*] [-k] [-n *number*] [-s] *file args*

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The csplit utility reads *file* and separates it into sections, one section for each of the *args* arguments. By default, the pieces are placed in files named xx00, xx01, ..., xx*n*, where *n* is 99, by default. These sections get the following pieces of *file*:

00: From the start of *file* up to (but not including) the line referenced by *arg1*.
01: From the line referenced by *arg1* up to the line referenced by *arg2*.
.
.
.
n+1: From the line referenced by *argn* to the end of *file*.

The csplit utility accepts the following options and operands:

-f *prefix* Specifies the created files *prefix*00, *prefix*01, ..., *prefixn*. The default is xx00, xx01, ..., xx*n*. If the *prefix* argument would create a file name that exceeds {NAME_MAX} bytes, an error occurs and csplit exits.

-k Leaves previously created files intact. By default, csplit removes created files if an error occurs.

-n *number* Uses *number* decimal digits to form file names for the file pieces. The default is 2.

-s Suppresses the printing of all character counts. The csplit utility usually prints the character counts for each file created.

file The path name of a text file to be split. If *file* is -, the standard input is used.

args The *args* operands can be a combination of the following:

/*rexp*/[*offset*]	Creates a file for the piece from the current line up to (but not including) the line that contains the regular expression *rexp*. The optional *offset* is a positive or negative integer value preceded by a + or -. After the section is created, the current line is set to the line that results from the evaluation of the regular expression with any offset applied.
%*rexp*%[*offset*]	This argument is the same as /*rexp*/, except that no file is created for the section.
line_no	A file is to be created from the current line up to (but not including) line number, *line_no*. The current line becomes *line_no*.
{*num*}	Repeat argument. This argument may follow any of the preceding arguments. If it follows a *rexp* type argument, that argument is applied *num* more times. If it follows *lnno*, the file will be split every *lnno* lines (*num* times) from that point.

Enclose all *rexp* type arguments that contain blanks or other characters meaningful to a shell in the appropriate quotation marks. Regular expressions may not contain embedded new lines. `csplit` does not affect the original file; the user must remove it.

NOTES

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
`system`, `secadm`	Allowed to split any file. In a privileged administrator shell environment, shell-redirected I/O is not subject to file protections.
`sysadm`	Allowed to split any file subject to security label restrictions. Shell-redirected I/O is subject to security label restrictions.

If the `PRIV_SU` configuration option is enabled, the super user is allowed to split any file. Shell-redirected I/O on behalf of the super user is not subject to file protections.

EXIT STATUS

The `csplit` utility exits with one of the following values:

0 Successful completion.

>0 An error occurred.

MESSAGES

Diagnostic messages are self-explanatory, except for the following:

```
arg - out of range
```

This message means that the given argument did not reference a line between the current position and the end of the file.

EXAMPLES

Example 1: This example creates four files, `fort00` ... `fort03`:

```
csplit -f fort file.f '/subroutine xyz/' '/function abc/' '/blockdata/'
```

After editing the ''split'' files, they can be recombined, as follows:

```
cat fort0[0-3] > file.f
```

This example overwrites the original file.

Example 2: This example splits the file at every 100 lines, up to 10,000 lines. The `-k` option causes the created files to be retained if less than 10,000 lines exist; however, an error message is still printed.

```
csplit -k file '100' '{99}'
```

Example 3: Assuming that `prog.c` follows the typical `C` coding convention of ending routines with a `}` at the beginning of the line, this example creates a file that contains each separate C routine (up to 21) in `prog.c`.

```
csplit -k prog.c '%main(%' '/^}/+1' '{20}'
```

Example 4: This example creates up to 20 chapter files from the file `novel`:

```
csplit -k -f chap. novel '%CHAPTER%' '{20}'
```

SEE ALSO

`ed`(1), `sh`(1), `split`(1)

NAME

ctags – Creates a tags file

SYNOPSIS

```
ctags [-a] [-f tagsfile] [-t] [-u] [-w] [-B] [-F] pathname ...
ctags -v [-t] [-w] pathname ...
ctags -x [-t] [-w] pathname ...
```

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4
AT&T extensions (-t, -u, -v, -w, -B, and -F options)

DESCRIPTION

The ctags utility makes a tags file for ex(1) from the specified C, Pascal, Fortran, yacc, lex, and Lisp sources. A tags file provides the locations of specified objects (in this case, functions and typedefs) in a group of files. Each line of the tags file contains the object name, the file in which it is defined, and an address specification for the object definition. Functions are searched with a pattern, typedefs with a line number. Specifiers are entered in separate fields on the line, separated by <blank>s or <tab>s. Using the tags file, ex(1) can quickly find these object definitions.

Files that have names that end in .c or .h are assumed to be C source files and are searched for C routine and macro definitions. Files that have names that end in .y are assumed to be yacc source files. Files whose names end in .l are assumed to be either Lisp files (if their first non-<blank> character is ;, (, or [), or lex files (if the first non-<blank> character is anything else). Other files are first examined to see whether they contain any Pascal or Fortran routine definitions; if they do not, they will be processed again for C definitions.

The ctags utility accepts the following options:

-a Appends to tags file.

-f *tagsfile* Tag descriptions are placed in *tagsfile*. By default, they are placed in file tags.

-t Creates tags for typedefs.

-u Updates the specified files in *tagsfile*; that is, all references to them are deleted, and the new values are appended to the file.
Note: It is usually faster to rebuild the *tagsfile* file.

-v Produces an index on the standard output. This listing contains the function name, file name, and page number (assuming 64 line pages). Because the output will be sorted into lexicographic order, you may want to run the output through sort -f. A sample follows:

```
ctags -v files | sort -f > index
```

`-w` Suppresses warning diagnostics.

`-x` Produces a list of object names, the line number, and the file name on which each is defined, as well as the text of that line, and it prints this on standard output. This is a simple index that can be printed out as an offline-readable function index.

`-B` Uses backward searching patterns (`?...?`).

`-F` Uses forward searching patterns (`/.../`) (default).

The `main` tag is treated in a special way in C programs. The tag formed is created by prepending of `M` to the name of the file, with a trailing `.c`, if any, removed, and leading path name components also removed. This makes the use of `ctags` practical in directories that have more than one program.

EXIT STATUS

The `ctags` utility exits with one of the following values:

0 Successful completion.

>0 An error occurred.

BUGS

Procedures cannot have the same name even if they are in different blocks.

`#ifdef`s are not recognized.

`ctags` relies on well-formed input to detect `typedef`s. The use of the `-tx` options shows only the last line of `typedef`s.

`ctags` does not recognize Pascal types.

FILES

`tags` Output tags file

SEE ALSO

`ex`(1), `vi`(1)

NAME

`cut` – Cuts out selected fields of each line of a file

SYNOPSIS

```
cut -b list [-n] [file...]
cut -c list [file...]
cut -f list [-d delim] [-s] [file...]
```

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The `cut` utility cuts out columns from a table or fields from each line of a file; in database parlance, it implements the projection of a relation. The fields as specified by *list* can be of fixed length, that is, character positions as on a punched card (`-c` option), or the length can vary from line to line and be marked with a field delimiter character such as `<tab>` (`-f` option). The `cut` utility can be used as a filter; if no files are specified, the standard input is used. In addition, a file name of `-` explicitly refers to standard input.

The `cut` utility accepts the following options:

`-b` *list* — The *list* following `-b` specifies byte positions. For example, `-b 8-56` passes all bytes that start at byte 8 through and including byte 56 of each line.

`-c` *list* — The *list* following `-c` specifies character positions. For example, `-c 1-72` passes the first 72 characters of each line.

`-d` *delim* — The character following `-d` is the field delimiter (`-f` option only). Default is the `<tab>` character. You must enclose in quotation marks spaces or other characters that have special meaning to the shell.

`-f` *list* — The *list* following `-f` is a list of fields assumed to be separated in the file by a delimiter character (see `-d`). For example, `-f 1,7` only copies the first and seventh fields. Lines with no field delimiters are passed through intact (useful for table subheadings) unless you specify `-s`.

`-n` — When specified with the `-b` option, `-n` prevents characters which are made up of more than one byte from being split. (In UNICOS 8.0 release, multibyte characters are not supported.)

`-s` — Suppresses lines with no delimiter characters in the case of the `-f` option. Unless specified, lines with no delimiters are passed through untouched.

file — A path name of a file (any type) to be used by this command.

list A list, separated by commas or blank characters, of integer field numbers (in increasing order), with optional `-` to indicate ranges. For example, `1,4,7`; `1-3,8`; `-5,10` (short for `1-5,10`); or `3-` (short for third through last field.)

You must specify either the `-b`, `-c` or the `-f`.

Use `grep`(1) to make horizontal ''cuts'' (by context) through a file, or use `paste`(1) to put files together by columns (that is, horizontally). To reorder columns in a table, use `cut` and `paste`.

NOTES

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
`system`, `secadm`	In a privileged administrator shell environment, shell-redirected I/O is not subject to file protections.
`sysadm`	Shell-redirected output is subject to security label restrictions.

If the `PRIV_SU` configuration option is enabled, shell-redirected I/O on behalf of the super user is not subject to file protections.

EXIT STATUS

The `cut` utility exits with one of the following values:

0 All input files were output successfully.

>0 An error occurred.

MESSAGES

`ERROR: line too long`
A line can have no more than 1022 characters or fields, or no newline character exists.

`ERROR: bad list for c/f option`
Missing `-c` or `-f` option or incorrectly specified *list*. If a line has fewer fields than the *list* indicates, no error occurs.

`ERROR: no fields`
The *list* is empty.

`ERROR: no delimiter`
Missing *delimiter* on `-d` option.

`ERROR: cannot handle multiple adjacent backspaces`
Adjacent backspaces cannot be processed correctly.

`WARNING: cannot open <filename>`
Either *filename* cannot be read or it does not exist. If multiple file names are present, processing continues.

EXAMPLES

Example 1: This example shows the mapping of user IDs to names:

```
cut -d: -f 1,5 /etc/passwd
```

Example 2: This example shows your login name:

```
name=‘who am i | cut -f 1 -d" "‘
```

SEE ALSO

`grep`(1), `paste`(1)

NAME

cxref – Generates C-language program cross-reference table

SYNOPSIS

cxref [-C] [-c] [-d] [-D *name* [*-def*]]... [-F] [-I *dir*] [-l] [-L *cols*] [-o *file*] [-s] [-t] [-U *dir*]... [-V] [-w *num*] [-W *name, file, function, line*] *files*

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

AT&T (-d, -l, -C, -F, -L, -V, -W)

DESCRIPTION

The cxref utility analyzes a collection of C files and builds a cross-reference table. cxref uses a special version of cc(1) to include #define'd information in its symbol table. It generates a list of all symbols (auto, static, and global) in each individual file, or, with the -c option, in combination. The table includes four fields: NAME, FILE, FUNCTION, and LINE. The line numbers appearing in the LINE field also show reference marks as appropriate. The reference marks include the following:

=	assignment
-	declaration
*	definition

If no reference marks appear, you can assume a general reference.

The cxref utility interprets the -D, -I, -U options in the same manner that cc(1) does. In addition, cxref interprets the following options:

-c Combines the source files into a single report. Without the -c option, cxref generates a separate report for each file on the command line.

-C Runs only the first pass of cxref, creating a .cx file that can later be passed to cxref. This is similar to the -c option of cc(1) or lint(1).

-d Disables printing declarations, making the report easier to read.

-F Prints the full path of the referenced file names.

-l Does not print local variables. Prints only global and file scope statistics.

-L *cols* Modifies the number of columns in the LINE field. If you do not specify a number, cxref defaults to five columns.

-o *file* Direct output to *file*.

`-s` Operates silently; does not print input file names.

`-t` Formats listing for 80-column width.

`-V` Prints version information on the standard error.

`-w` *num* Width option that formats output no wider than *num* (decimal) columns. This option will default to 80 if *num* is not specified or is less than 51.

`-W` *name, file, function, line*
Changes the default width of at least one field. The default widths are as follows:

Field	Characters
NAME	15
FILE	13
FUNCTION	15
LINE	20 (4 per column)

DIAGNOSTICS

Error messages usually mean you cannot compile the files.

EXIT STATUS

The `cxref` utility exits with one of the following values:

0 Successful completion.

>0 An error occurred.

EXAMPLES

```
$ nl a.c
1       main()
2       {
3             int i;
4               extern char c;
5
6               i=65;
7               c=(char)i;
8       }
```

```
NAME    FILE          FUNCTION   LINE
c       a.c           ---        4-    7=
i       a.c           main       3*    6=   7
```

NAME	FILE	FUNCTION	LINE
main	a.c	---	2*
u3b2	Predefined	---	0*
unix	Predefined	---	0*

FILES

TMPDIR/tcx.*	Temporary files
TMPDIR/cx.*	Temporary files
LIBDIR/xref	Accessed by cxref
LIBDIR	/usr/lib by default
TMPDIR	Usually /var/tmp but can be redefined by setting the environment variable TMPDIR (see tempnam in tmpnam(3C)).

SEE ALSO

cc(1), lint(1)

tmpnam(3C) in the *UNICOS System Libraries Reference Manual*, Cray Research publication SR–2080

NAME

`date` – Prints and sets the date

SYNOPSIS

```
date [-u] [+format]
date [-b] [-u] mmddhhmm[[cc]yy]
date -a [-]sss.fff
```

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4
AT&T extension (`-a` option)
CRI extension (`-b` option)

DESCRIPTION

If you omit arguments, or if the argument begins with `+`, the current date and time are printed; otherwise, the current date is set (only by super user).

The `date` utility accepts the following options:

`-a` [`-`] *sss.fff*
: Slowly adjusts the time by *sss.fff* seconds (*fff* represents fractions of a second). This adjustment can be positive or negative. The system's clock will be sped up (positive) or slowed down (negative) until it has drifted by the number of seconds specified.

`-b`
: Indicates that this is a system boot; therefore, the date change records written to `/etc/wtmp` should not be written. You should not call `date` with this option at any other time.

`-u`
: Displays (or sets) the date in Greenwich mean time (GMT-universal time), bypassing the normal conversion to (or from) local time.

  mm Month

  dd Day

  HH Hour (24-hour system)

  MM Minute

  cc Century minus one

  yy The last 2 digits of the year

  The month, day, year, and century may be omitted; the current values are supplied as defaults. The following example sets the date to October 8, 12:45 A.M.:

```
date 10080045
```

The current year is the default because no year is supplied. The system operates in GMT. `date` converts to and from local standard and daylight time. Only an appropriately authorized user may change the date. After successfully setting the date and time, `date` displays the new date according to the default format. The `date` utility uses `TZ` to determine the correct time zone information (see `environ`(7)).

+ *format* If the argument begins with +, the output of `date` is under the user's control. Each Field Descriptor (a description follows) is preceded by % and is replaced in the output by its corresponding value. One % is encoded by %%. All other characters are copied to the output without change. The string is always terminated with a <newline> character. If the argument contains embedded blanks, you must enclose it within single quotation marks (see the EXAMPLE section).

Specifications of native language translations of month and week day names are supported. The month and week day names used for a language are based on the locale specified by the `LC_TIME` and `LANG` environment variables (see `environ`(7)).

The month and week day names used for a language are taken from a file whose format is specified in `strftime`(3C). This file also defines country-specific date and time formats such as `%c`, which specifies the default date format. The following form is the default for `%c`:

```
%a %b %e %T %Z %Y
```

for example,

```
Fri Dec 23 10:10:42 EST 1988
```

Field Descriptors

Field descriptors must be preceded by a %:

a Abbreviated week day name

A Full week day name

b Abbreviated month name

B Full month name

c Country-specific date and time format

C Century (a year divided by 100 and truncated to an integer) as a decimal number – 00 through 99

d Day of month – 01 through 31

D Date as `%m/%d/%y`

e Day of month – 1 through 31 (single digits are preceded by a blank)

h Abbreviated month name (alias for `%b`)

H Hour – 00 through 23

- `I` Hour – 01 through 12
- `j` Day of year – 001 through 366
- `m` Month of year – 01 through 12
- `M` Minute – 00 through 59
- `n` Insert a `<newline>` character
- `p` String containing ante-meridiem or post-meridiem indicator (by default, AM or PM)
- `r` Time as `%I:%M:%S %p`
- `R` Time as `%H:%M`
- `S` Second – 00 through 61, allows for leap seconds
- `t` Insert a `<tab>` character
- `T` 24h clock time as `%H:%M:%S`
- `u` Week day as a decimal number – Monday = 1
- `U` Week number of year (Sunday as the first day of the week) – 00 through 53
- `V` Week of the year (Monday as the first day of the week) – 01 through 53
- `w` Day of week – Sunday = 0
- `W` Week number of year (Monday as the first day of the week) – 00 through 53
- `x` Country-specific date format
- `X` Country-specific time format
- `y` Year within century – 00 through 99
- `Y` Year as *ccyy* (4 digits)
- `Z` Time zone name

Modified Field Descriptors

Modified field descriptors must be preceded by a `%`. You can modify some field descriptors by the `E` and `O` modifier characters to indicate a different format or specification as specified in the `LC_TIME` locale description. If the corresponding keyword is not specified or not supported for the current locale, the unmodified field descriptor value is used. (In the UNICOS 9.0 release, alternate locale representations are not supported.)

- `Ec` Alternate appropriate date and time representation
- `EC` The name of the base year (period) in the alternate representation
- `Ex` Alternate date representation
- `Ey` Offset from `%EC` (year only) in the alternate representation
- `EY` Full alternate year representation

`Od` Day of the month using the alternate numeric symbols

`Oe` Day of the month using the alternate numeric symbols

`OH` Hour (24h clock) of the month using the alternate numeric symbols

`OI` Hour (12h clock) of the month using the alternate numeric symbols

`Om` Month using the alternate numeric symbols

`OM` Minutes using the alternate numeric symbols

`OS` Seconds using the alternate numeric symbols

`Ou` Week day as a number in the alternate representation – Monday = 1

`OU` Week number of the year (Sunday as the first day of the week) using the alternate numeric symbols

`OV` Week number of the year (Monday as the first day of the week) using the alternate numeric symbols

`Ow` Week day as a number in the alternate representation (Sunday = 0)

`OW` Week number of the year (Monday as the first day of the week) using the alternate numeric symbols

`Oy` Year (offset from `%C`) in alternate representation

NOTES

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
`system`, `secadm`	Allowed to set the date.
`sysadm`, `sysops`	Allowed to set the date. Shell-redirected I/O is subject to security label restrictions.

If the `PRIV_SU` configuration option is enabled, the super user is allowed to set the date.

If you try to set the current date to one of the dates that the standard and alternate time zones change (for example, the date that daylight time is starting or ending), and you try to set the time to a time in the interval between the end of standard time and the beginning of the alternate time (or the end of the alternate time and the beginning of standard time), the results are unpredictable.

EXIT STATUS

The `date` utility exits with one of the following values:

0 The date was written successfully.

>0 An error occurred.

MESSAGES

`No permission` You are not authorized to set the date.

`bad conversion` The date set is syntactically incorrect.

EXAMPLES

The following command:

```
date '+DATE: %m/%d/%y%nTIME: %H:%M:%S'
```

generates the following output:

```
DATE: 08/01/76
TIME: 14:45:05
```

SEE ALSO

`adjtime`(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

`printf`(3C), `strftime`(3C) in the *UNICOS System Libraries Reference Manual*, Cray Research publication SR–2080

`environ`(7) (available only online)

`setdate`(8) in the *UNICOS Administrator Commands Reference Manual*, Cray Research publication SR–2022

General UNICOS System Administration, Cray Research publication SG–2301

NAME

dd – Converts and copies a file to the specified output device

SYNOPSIS

dd [*operand...*]

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4
AT&T extension (iseek and oseek operands)

DESCRIPTION

The dd utility copies the specified input file to the specified output device with possible conversions. Standard input and output are used by default. You can specify the input and output block size to take advantage of raw physical I/O. *operand* takes the form option=*value*.

Operands are as follows:

bs=*n* Sets both input and output block size, superseding ibs and obs. When you do not specify a conversion, it is particularly efficient because an in-core copy does not have to be done.

cbs=*n* Specifies the conversion block size for block and unblock in bytes by *expr* (the default is 0). If cbs=*n* is omitted or is 0, using lock or unblock produces unspecified results. The cbs operand is used only when ascii or ebcdic conversion is specified. In the former case, cbs characters are placed in the conversion buffer and converted to ASCII, trailing blanks are trimmed, and <newline>s are added before the line is sent to output. In the latter case, ASCII characters are read into the conversion buffer and converted to EBCDIC, and blanks are added to make up an output block of size cbs.

conv=*value*[,*value...*]
Performs the specified conversion; *value* are comma-separated symbols from the following list:

ascii	Converts EBCDIC to ASCII.
block	Converts <newline>-terminated ASCII records to fixed length.
ebcdic	Converts ASCII to EBCDIC.
ibm	Maps ASCII to EBCDIC in a slightly different way.
lcase	Maps alphabetic characters to lowercase.
noerror	Does not stop processing on an error.

`notrunc`	Does not truncate the output file. Preserves blocks in the output file not explicitly written by this invocation of the `dd` utility.
`swab`	Swaps every pair of bytes.
`ucase`	Maps alphabetic characters to uppercase.
`unblock`	Converts fixed-length ASCII records to `<newline>`-terminated records.
`sync`	Pads every input block to `ibs`.

`count=`*n* Copies only *n* blocks.

`files=`*n* Copies and concatenates *n* input files before terminating. Using this operand makes sense only when input is a magnetic tape or similar device.

`ibs=`*n* Specifies input block size of *n* bytes. Default is 512.

`if=`*file* Specifies input file name. Standard input is default.

`iseek=`*n* Seeks *n* input blocks from beginning of input file before starting to copy.

`obs=`*n* Specifies output block size of *n* bytes. Default is 512.

`of=`*file* Specifies output file name. Standard output is default. The `dd` utility creates an explicit output file; therefore, the `seek` option is usually useless with an explicit output file except in special cases, such as when using disk files.

`seek=`*n*
`oseek=`*n* Seeks *n* output blocks from beginning of output file before copying. This option generally works only with raw disk files and does not work when the explicit output file was specified using the `of` option.

`skip=`*n* Skips *n* input blocks before starting copy. The skipped blocks are actually read; therefore, this can take a considerable amount of time.

When sizes are specified, a number of bytes is expected. A number may end with `b`, `s`, `k`, `B`, or `w` to specify multiplication by 512, 512, 1024, 4096, or 8, respectively; a pair of numbers may be separated by `x` to indicate a product.

After completion, `dd` reports the number of whole and partial input and output blocks.

NOTES

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	**Action**
`system`, `secadm`	Allowed to copy any file. In a privileged administrator shell environment, shell-redirected I/O is not subject to file protections.
`sysadm`	Allowed to copy any file subject to security label restrictions. Shell-redirected I/O is subject to security label restrictions.

If the PRIV_SU configuration option is enabled, the super user is allowed to copy any file. Shell-redirected I/O on behalf of the super user is not subject to file protections.

EXIT STATUS

The dd utility exits with one of the following values:

0 The input file was copied successfully.

>0 An error occurred.

MESSAGES

f+p records in(out) Numbers of full and partial records read (written)

EXAMPLES

Example 1: The following command reads an EBCDIC file that is blocked, ten 80-byte EBCDIC card images per block, into ASCII file x:

```
dd if=file of=x ibs=800 cbs=80 conv=ascii,lcase
```

Example 2: The dd utility is especially suited to I/O on raw physical devices because it permits reading and writing in arbitrary block sizes.

To use dd to copy the myfile file in your home directory out to a tape with the volume serial number of UA1234, enter the following:

```
rsv
tpmnt -l nl -p /tmp/$$u -v UA1234 -b 32768
dd if=$HOME/myfile of=/tmp/$$u bs=32768
rls -a
```

To read this file back in from tape to a disk file called newfile in your home directory, enter the following:

```
rsv
tpmnt -l nl -p /tmp/$$u -v UA1234 -b 32768
dd if=/tmp/$$u of=$HOME/newfile bs=32768
rls -a
```

The block size matches the maximum block size in the tpmnt(1) command.

SEE ALSO

cp(1), rls(1), rsv(1), tpmnt(1)

NAME

`define` – Displays definitions of Cray Research technical terms and terms added by a local site that match a specified search term

SYNOPSIS

```
define searchterm
define -a
define [-k] searchterm
define -l
define [-x] searchterm
```

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `define` utility displays definitions of Cray Research technical terms and terms added by a local site that match a specified search term (*searchterm*). If you enter `define` *searchterm*, `define` displays all matched terms and associated definitions. If `define` finds an exact match for *searchterm*, `define` displays the exact match and its definition first, followed by all other possible matches. If an exact match is not found, `define` displays any other possible matches and definitions.

If a term contains the specified *searchterm* string, `define` considers it a match. For example, entering `define uid` displays all terms (and their definitions) that contain the string `uid` (such as `setuid` and `subcooled liquid`).

Always specify the desired search term in lowercase characters (for example, enter `define ascii`, not `define ASCII`).

When using any of the following special or wildcard characters within a *searchterm*, protect the characters by using single quotation marks. The special or wildcard characters include:

```
$  *  [  ^  |  (  )  +  /
```

For example, enter `define 'sys$input'`, not `define sys$input`; or enter `define 'net*'`, not `define net*`.

The `define` utility accepts multiple word search terms (for example, you may specify `define character set` on the command line; `character set` is the search term).

The `define` utility accepts the following options and arguments:

`-a` Displays the entire glossary (all available terms and definitions) in alphabetical order.

`-k` When you use with *searchterm*, displays all matched terms (without the definitions). For example, entering `define -k check` retrieves an alphabetical list of terms that match the *searchterm* `check` (such as `check bits (cb)`, `check digit, checkbyte`, `checkpoint`, `checksum`, `cyclic redundancy check (crc)`, `debug (troubleshoot) (checkout)`, and `parity check`).

`-l` Displays in alphabetical order all available terms, synonyms, and acronyms (without the definitions). Synonyms and acronyms are displayed within parentheses next to the term. The list of terms produced by the `-l` option appears in all lowercase characters; however, when the `define` utility accesses a term, it will appear in the correct case.

`-x` When used with the *searchterm* argument, displays only a term (and its associated definition) that matches the *searchterm* exactly. For example, entering `define -x file` displays the definition for only the term `file`.

searchterm
Specifies the term for which to search in the online glossary.

If the output device is a terminal, `define` pipes its output through the pager specified by the `PAGER` environment variable. If you omit `PAGER`, `define` uses `more -s` as the default pager. If the output device is not a terminal, `define` sends output to the standard output device (`stdout`).

NOTES

Generally, the `define` utility's glossary does not contain definitions for UNICOS commands or standard UNIX terms. To obtain information on UNICOS commands, use the `man`(1) command. (For example, to find out the function of the `more` command, enter`man more`).

The default definitions directory that `define` uses is `/usr/lib/define`, unless you set the `DEFINEDIR` environment variable. The `define` utility reads files that have a `_k` suffix in the definitions directory as keyword files and corresponding files without the `_k` suffix as definition files. Each keyword file in the definitions directory is checked for validity. If a keyword file is not valid, its corresponding definition file will not be used. `define` searches each definition file in the directory in sequence.

For information on modifying the Cray Research definitions file or creating a local definitions file, see `builddefs`(1).

FILES

`CRAYdefs`	Cray formatted definition file
`CRAYdefs_i`	Cray formatted definitions with keywords embedded
`CRAYdefs_k`	Cray keyword file (non-ASCII)
`CRAYdefs_src`	Cray unformatted source definitions
`define.cat`	The `define` message catalog
`define.exp`	The `define` explain message catalog

`define.msg` The `define` message text file

SEE ALSO

`builddefs`(1), `man`(1)

General UNICOS System Administration, Cray Research publication SG–2301

NAME

delta – Makes a delta (change) to an SCCS file

SYNOPSIS

delta [-g *list*] [-m *mrlist*] [-n] [-p] [-r *SID*] [-s] [-y[*comment*]] *files*

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The delta utility is used to permanently introduce into the named Source Code Control System (SCCS) file changes that were made to the file retrieved by get(1) (called the *g-file*, or generated file).

The delta utility makes a delta to each named SCCS file. If a directory is named, delta behaves as though each file in the directory were specified as a named file, except that non-SCCS files (last component of the path name does not begin with s.) and unreadable files are silently ignored. If a dash (-) is given as a name, the standard input is read (see the WARNINGS section); each line of the standard input is taken to be the name of an SCCS file to be processed.

The delta utility may issue prompts on the standard output depending upon certain options specified and flags (see admin(1)) that may be present in the SCCS file (see -m and -y options).

Option arguments apply independently to each named file.

-g *list* Specifies a *list* (see get(1) for the definition of *list*) of deltas that are to be ignored when the file is accessed at the change level (SID) created by this delta.

-m *mrlist* If the SCCS file has the v flag set (see admin(1)), a modification request (MR) number *must* be supplied as the reason for creating the new delta.

-n Specifies retention of the edited *g-file* (normally removed at completion of delta processing).

-p Causes delta to print (on the standard output) the SCCS file differences before and after the delta is applied in a diff(1) format.

-r *SID* Uniquely identifies which delta is to be made to the SCCS file. The use of this option is necessary only if two or more outstanding gets for editing (get -e) on the same SCCS file were done by the same person (login name). The *SID* value specified with the -r option can be either the source identifier (SID) specified on the get command line or the SID to be made as reported by the get command (see get(1)). A message results if the specified SID is ambiguous, or if necessary and omitted on the command line.

`-s` Suppresses the issue, on the standard output, of the created delta's SID, as well as the number of lines inserted, deleted, and unchanged in the SCCS file.

If `-m` is not used and the standard input is a terminal, the prompt `MRs?` is issued on the standard output before the standard input is read; if the standard input is not a terminal, a prompt is not issued. The `MRs?` prompt always precedes the `comments?` prompt (see `-y` option).

MRs in a list are separated by blanks and/or tab characters. An unescaped new-line character terminates the MR list.

If the `v` flag has a value (see `admin`(1)), it is taken to be the name of a program (or shell procedure), that will validate the correctness of the MR numbers. If a nonzero exit status is returned from MR number validation program, `delta` terminates (it is assumed that the MR numbers were not all valid).

`-y`[*comment*] Arbitrary text used to describe the reason for making the delta. A null string is considered a valid *comment*.

If `-y` is not specified and the standard input is a terminal, the prompt `comments?` is issued on the standard output before the standard input is read; if the standard input is not a terminal, a prompt is not issued. An unescaped new-line character terminates the comment text.

files Specifies the SCCS files to change.

WARNINGS

Lines beginning with an `SOH` ASCII character (binary 001) cannot be placed in the SCCS file, unless the `SOH` is escaped. This character has special meaning to SCCS (see `sccsfile`(5)) and will cause an error.

A `get` of many SCCS files, followed by a `delta` of those files, should be avoided when the `get` generates a large amount of data. Instead, multiple `get/delta` sequences should be used.

If the standard input (`-`) is specified on the `delta` command line, the `-m` (if necessary) and `-y` options must also be present. If you omit these options, an error occurs.

Comments are limited to text strings of 512 characters.

MESSAGES

Use `help`(1) for explanations.

EXAMPLES

In the following example, changes found in the file `example.c` are applied to `s.example.c`. The file `example.c` is removed. User input is shown in bold type:

```
$ delta s.example.c
comments? Changed some message text.
1.2
1 inserted
1 deleted
4 unchanged
$
```

FILES

g-file	Existed before the execution of `delta`; removed after completion of `delta`.
p-file	Existed before the execution of `delta`; may exist after completion of `delta`.
q-file	Created during the execution of `delta`; removed after completion of `delta`.
x-file	Created during the execution of `delta`; renamed to SCCS file after completion of `delta`.
z-file	Created during the execution of `delta`; removed during the execution of `delta`.
d-file	Created during the execution of `delta`; removed after completion of `delta`.
`/usr/bin/bdiff`	Program to compute differences between the ‘‘gotten’’ file and the *g-file*.

SEE ALSO

`admin`(1), `bdiff`(1), `cdc`(1), `comb`(1), `diff`(1), `get`(1), `help`(1), `prs`(1), `rmdel`(1), `sact`(1), `sccsdiff`(1), `unget`(1), `val`(1), `vc`(1), `what`(1)

`chown`(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

`sccsfile`(5) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

NAME

deplib – Manipulates dependent library names in bld library files

SYNOPSIS

deplib [-l] [-a *file...*] [-d *file...*] [-r *file...*] *bld_library*

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The deplib utility lets users associate library names with bld(1) libraries. With deplib, users can list, add, delete, and replace dependent library names in the bld library file headers. If dependent library names are present within a bld library, segldr(1) searches those libraries immediately after searching the library in which they are specified. For example, if the user specifies libraries A, B, and C, and B declares dependent libraries D and E, the search order is A, B, D, E, C followed by the default libraries.

Dependent libraries are only recognized by segldr (ld(1) will ignore them). Also, deplib will operate only on bld libraries, not on ar(1) libraries.

The deplib utility accepts the following options:

-l Lists any dependent library names associated with *bld_library*.

-a *file* Adds the file or files to the dependent library list in *bld_library*. If more than one file is to be added, enclose the list in single quotation marks or separate the list with commas.

-d *file* Deletes the file or files from the dependent library list in *bld_library*. If more than one file is to be deleted, enclose the list in single quotation marks or separate the list with commas.

-r *file* Replaces the current list of dependent library names with the list given on the command line. Note that this is a total replacement of filenames. If more than one file is in the replacement list, enclose the list in single quotation marks or separate the list with commas.

bld_library Specifies the name of the bld library.

Dependent library names can be either a full or relative path name or just a file name. If the string begins with a . or /, segldr assumes the string is a complete path name. If the string begins with a ., the current directory is used, which is not necessarily the directory in which the bld library resides. Otherwise, segldr assumes that the string is only a file name, with no path. In this case, segldr will look in each directory specified by -L (on the segldr command line) for the file and take the first one found.

NOTES

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
system, secadm	Allowed to manipulate any file. In a privileged administrator shell environment, shell-redirected I/O is not subject to file protections.
sysadm	Allowed to manipulate any file subject to security label restrictions. Shell-redirected I/O is subject to security label restrictions.

If the PRIV_SU configuration option is enabled, the super user is allowed to manipulate any file. Shell-redirected I/O on behalf of the super user is not subject to file protections.

EXAMPLES

Example 1: The following example adds two library names to the bld library, mylib.a:

```
$ deplib -a 'mylibf.a mylibm.a' mylib.a
```

Example 2: The following example lists the current dependent libraries in mylib.a, replaces the list, and lists them again:

```
$ deplib -l mylib.a
mylibf.a
mylibm.a
$ deplib -r 'yourlibf.a yourlibm.a' mylib.a
$ deplib -l mylib.a
yourlibf.a
yourlibm.a
```

SEE ALSO

bld(1), segldr(1)

NAME

deroff – Removes nroff, troff, tbl, and eqn constructs

SYNOPSIS

deroff [-w] *filenames*

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The deroff utility reads each file in sequence and removes all nroff(1) and troff(1) command lines, backslash constructions, macro definitions, eqn constructs (between .EQ and .EN lines or between delimiters), and table descriptions and writes the remainder on the standard output. deroff follows chains of included files (.so and .nx commands); if a file has already been included, a .so is ignored and a .nx terminates execution. If no input file is given, deroff reads from the standard input file.

The deroff utility accepts the following option and operand:

-w Generates a word list, one word per line. A *word* is a string of letters, digits, and apostrophes, beginning with a letter; apostrophes are removed. All other characters are ignored.

filenames Specifies the files on which to run deroff.

NOTES

The deroff utility is not a complete troff(1) interpreter, so it can be confused by subtle constructs. Most errors result in too much rather than too little output.

The deroff utility does not work well with files that use .so to source in the standard macro package files.

SEE ALSO

eqn(1), nroff(1), tbl(1), troff(1)

NAME

df – Reports free disk space

SYNOPSIS

```
df [-S] [-d] [-f] [-l] [-p] [-P | -t] [file ...]
df -B [-d] [-f] [-l] [-p] [-P | -t] [file ...]
df -k [-d] [-f] [-l] [-p] [-P | -t] [file ...]
```

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4
CRI extensions (-B, -d, -f, -l, -p, -S, and -t options)

DESCRIPTION

The df utility reports the number and percent of free blocks available for file systems by examining the counts kept in the super blocks; you can specify *file* by device name (such as /dev/dsk/root), by mounted directory name (such as /usr), or by the pathname of a file that exists on a file system. If you do not specify the *file* operand, the free space on all of the mounted file systems is printed. The default output reports the following information in order: file name, device name, number of free blocks, the block size in bytes, free percentage, and, if appropriate, the number of free file items. The items may be *inodes* or *processes* (/proc). If no tracks remain on the file system, an asterisk (*) is displayed after the free percentage. The default block size when reporting space figures is 512-byte units. You can use the -B or -k options to override this default, showing instead the number of 4096-byte units and 1024-byte units, respectively.

The df utility accepts the following options and operands:

-B Reports number of blocks that in 4096-byte units, instead of the default 512-byte units.

-d Reports information on partitions which are flagged as DOWN, READ-ONLY, or NO-ALLOCATE. For mounted file systems, the default behavior of df is to not report usage information for partitions that are "offline." For unmounted file systems, this option has no effect. (For additional information on marking file systems as DOWN, READ-ONLY, or NO-ALLOCATE, see pddconf(8) for Cray Research systems having I/O model E subsystems (IOS-E).

-f Option provided for backward compatibility only; this option does nothing, and will be removed in the next release of UNICOS.

-k Reports number of blocks in 1024-byte units, instead of the default 512-byte units.

-l Reports only on local file systems. NFS file systems are omitted.

`-p` Reports the same information as the `-t` option (in sectors), thresholds for big file allocations, and allocation size for primary or secondary partitions, if greater than one block. It also reports the following information for each partition, in table form:

`part`	Relative partition number
`start`	Starting block number
`total`	Total block count of partition
`free`	Free block count of partition
`frags`	Count of discontiguous areas in the partition
`device`	ASCII name of the device on which the partition resides

For `NC1FS` file systems, the inode regions are also displayed.

`-P` Produces output in the format specified by POSIX 1003.2. The unit size used to report space figures depends on other options. By default, the figures are in 512-byte units. If you specify `-k`, the figures are in 1024-byte units; if you specify `-B`, the figures are in 4096-byte units. The output consists of a single header line followed by lines with the following fields:

file system name	Device name.
total space	Total size of file system in specified units.
space used	Total amount of space, in specified units, allocated to existing files in the file system.
space free	Total amount of space, in specified units, available within the file system.
percentage used	Percentage of the normally available space that is currently allocated to all files on the file system.
file system root	Directory below which the file system hierarchy appears.

`-S` Causes the output to be in a format easily modified by utilities, such as `sed`(1) or `cut`(1). The `-B` and `-k` options are ignored when this option is specified. The following information is displayed, delimited by the `<tab>` character:

file name	File system name.
device name	Device(s) on which the file system resides.
free blocks	Number of free blocks on the file system.
block size	The size of a block (in bytes).
free percentage	The ratio in percentage of free blocks to total blocks.
free item count	Count of free items. If items are not applicable to the file system, total items will be 0, and item type is `undef`.
total blocks	Total possible blocks on the file system.
total items	Total possible items.

item type A description of the item. Either `Inodes` for local file systems, `procs` for the `/proc` file system, or `undef` for file systems in which items are unavailable, or do not apply.

`-t` Includes in the output the total blocks and file items. Cannot be specified with the `-P` option.

file Name of the disk device.

NOTES

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

If the `PRIV_SU` configuration option is enabled, users are allowed to specify any file.

EXIT STATUS

The `df` utility exits with one of the following values:

0 Successful completion.

>0 An error occurred.

EXAMPLES

Example 1: The following is an example of information displayed by the `-t` and `-B` options:

```
file (/dev/dsk/scr05):   16580 4K blocks ( 33.6%) 1505 inodes
                 total:   49392 4K blocks          6600 inodes
```

Example 2: The following is an example of information displayed by the `-p` option only. This example shows an `NC1FS` file system, `/dev/dsk/qtest1`, that was constructed with the following command lines:

```
$ mkfs -q -P 4 -S 16 -p 1
$ df -p /dev/dsk/qtest1
```

```
qtest1     (/dev/dsk/qtest1  ):  23780 sectors       0 trks    6076 Inodes
                          total:  24192 sectors      (0 trks)   6080 Inodes

       Big file threshold:            32768 bytes
       Big file allocation minimum:      24 blocks

       Primary partitions allocation unit:         16K blocks
       Secondary partitions allocation unit:       64K blocks

       part   start     total         free (%)          frags (%)      device
       ----  --------  --------  -----------------  ----------------  --------
          0p        0      8064      7652 ( 94.9%)      0 (  0.000%)  50-A2-22
          1s     8064      8064      8064 (100.0%)      0 (  0.000%)  50-A1-22
          2s    16128      8064      8064 (100.0%)      0 (  0.000%)  50-A2-22

                              part: 0
          Inode region: 0                  Inode region: 1
        start    total  free    (%)      start    total  free    (%)
       --------  -----  -----  -----    --------  -----  -----  -----
              0   6080   6076  99.93
```

Example 3: The following is an example of the POSIX 1003.2 output format, using the `-P` and `-k` options.

```
$ df -P -k /dev/dsk/qtest

Filesystem        1024-blocks  Used       Available   Capacity  Mounted on
/dev/dsk/qtest    1096640      673776     422864       61%      qtest
```

FILES

`/dev/dsk/*`	Disk devices

SEE ALSO

`du`(1) to summarize disk usage

`statfs`(2) to get file system information

`dsk`(4) for information on disk drive, buffer memory, and SSD interface
`fs`(5) for file system partition format
in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

`pddconf`(8) to control the state of an IOS model E disk drive
in the *UNICOS Administrator Commands Reference Manual*, Cray Research publication SR–2022

General UNICOS System Administration, Cray Research publication SG–2301

NAME

diff – Differential file comparator

SYNOPSIS

diff [-bitw] [-c | -e | -f | -h | -n] *filename1 filename2*
diff [-bitw] [-c | -e | -f | -h | -n] [-l] [-r] [-s] [-S *name*] *directory1 directory2*
diff [-bitw] [-C *number*] *filename1 filename2*
diff [-bitw] [-C *number*] [-l] [-r] [-s] [-S *name*] *directory1 directory2*
diff [-bitw] [-D *string*] *filename1 filename2*

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4
AT&T extensions (-D, -i, -f, -h, -l, -n, -s, -S, -t, and -w options)

DESCRIPTION

The diff utility tells the lines that you must change in two files to bring them into agreement. If *filename1* (*filename2*) is -, the standard input is used. If *filename1* (*filename2*) is a directory, a file in that directory with the name *filename2* (*filename1*) is used. The typical output contains lines of these forms:

n1 a *n3,n4*
n1,n2 d *n3*
n1,n2 c *n3,n4*

These lines resemble ed(1) commands to convert *filename1* into *filename2*. The numbers after the letters pertain to *filename2*. In fact, by exchanging a for d and reading backward, you may ascertain equally how to convert *filename2* into *filename1*. As in ed(1), identical pairs, in which *n1 = n2* or *n3 = n4*, are abbreviated as one number.

Following each of these lines come all of the lines that are affected in the first file flagged by <, then all of the lines that are affected in the second file flagged by >.

The diff utility accepts the following options and operands:

-b Ignores trailing blanks (<space> and <tab> characters) and treats other strings of <blank>s as equivalent.

-i Ignores the case of letters (for example, 'A' will compare equal to 'a').

-t Expands <tab> characters in output lines. Typical or -c output adds character(s) to the front of each line that may adversely affect the indentation of the original source lines and make the output lines difficult to interpret. This option will preserve the original source's indentation.

`-w` Ignores all <blank>s (<space> and <tab> characters) and treats all other strings of <blank>s as equivalent. For example, '`if ( a == b )`' will compare equal to '`if(a==b)`'.

The following options are mutually exclusive:

`-c` Produces a listing of differences that has three lines of context. With this option, output format is modified slightly: output begins with identification of the files involved and their creation dates; then each change is separated by a line with a dozen `*`'s. The lines removed from *filename1* are marked with `-`; those added to *filename2* are marked `+`. Lines that are changed from one file to the other are marked with `!` in both files.

`-C` *number* Produces a listing of differences identical to that produced by `-c` with *number* lines of context.

`-e` Produces a script of *a*, *c*, and *d* commands for the editor `ed`(1), which will re-create *filename2* from *filename1*. In connection with `-e`, the following shell program may help maintain multiple versions of a file. Only an ancestral file (`$1`) and a chain of version-to-version `ed`(1) scripts (`$2`,`$3`,...) made by `diff` must be on hand. A ''latest version'' appears on the standard output.

```
(shift; cat $*; echo '1,$p') | ed - $1
```

Except in rare circumstances, `diff` finds a smallest sufficient set of file differences.

`-f` Produces a similar script, not useful with `ed`(1), in the opposite order.

`-h` Does a fast, half-hearted job. It works only when changed stretches are short and well separated, but it does work on files of unlimited length. Options `-e` and `-f` are unavailable with `-h`.

`-n` Produces a script similar to `-e`, but in the opposite order and with a count of changed lines on each insert or delete command.

`-D` *string* Creates a merged version of *filename1* and *filename2* with C preprocessor controls included so that a compilation of the result without defining *string* is equivalent to compiling *filename1*, while defining *string* yields *filename2*.

The following options are used for comparing directories:

`-l` Produces output in long format. Before the `diff`, each text file is piped through `pr`(1) to paginate it. Other differences are remembered and summarized after all text file differences are reported.

`-r` Applies `diff` recursively to common subdirectories encountered.

`-s` Reports files that are identical; these would not otherwise be mentioned.

`-S` *name* Starts a directory `diff` in the middle, beginning with the file *name*.

filename1
filename2 Path names of files to be compared.

directory1
directory2 Path names of directories to be compared.

NOTES

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
`system`, `secadm`	In a privileged administrator shell environment, shell-redirected I/O is not subject to file protections.
`sysadm`	Shell-redirected output is subject to security label restrictions.

If the `PRIV_SU` configuration option is enabled, shell-redirected I/O on behalf of the super user is not subject to file protections.

Editing scripts produced under the `-e` or `-f` option are naive about creating lines that consist of a one (`.`).

EXIT STATUS

The `diff` utility exits with one of the following values:

0 No differences were found.

1 Differences were found.

>1 An error occurred.

MESSAGES

`Missing newline at end of file` *X*
This message indicates that the last line of file *X* did not have a `<newline>` character. If the lines differ, they will be flagged and output; however, the output will seem to indicate that they are the same.

FILES

`TMPDIR/d`*tempfile*	Temporary working file
`/usr/lib/diffh`	Fast `diff` program used with the `-h` option
`/bin/pr`	`pr`(1) utility

SEE ALSO

`bdiff`(1), `cmp`(1), `comm`(1), `diff3`(1), `dircmp`(1), `ed`(1), `pr`(1),

`chown`(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

`group`(5), `passwd`(5), `udb`(5) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

NAME

diff3 – Makes a three-way differential file comparison

SYNOPSIS

```
diff3 [-e] file1 file2 file3
diff3 -x file1 file2 file3
diff3 -3 file1 file2 file3
```

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The diff3 utility compares three versions of a file, and it publishes disagreeing ranges of text flagged with these codes:

====	All three files differ.
====1	*file1* is different.
====2	*file2* is different.
====3	*file3* is different.

The type of change produced in the conversion of a given range of a given file to some other is indicated in one of the following ways:

f : *n1* a	Text is to be appended after line number *n1* in file *f*, *f* = 1, 2, or 3.
f : *n1* , *n2* c	Text is to be changed in the range line *n1* to line *n2*. If *n1* = *n2*, the range may be abbreviated to *n1*.

The original contents of the range follow immediately after a c indication. When the contents of two files are identical, the contents of the lower-numbered file are suppressed.

The diff3 utility accepts the following options:

-e Publishes for the ed(1) editor a script that incorporates into *file1* all changes between *file2* and *file3*, that is, the changes that normally would be flagged ==== and ====3.

-x Produces a script to incorporate only changes flagged ====.

-3 Produces a script to incorporate only changes flagged ====3.

NOTES

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
system, secadm	Allowed to compare any three files. In a privileged administrator shell environment, shell-redirected I/O is not subject to file protections.
sysadm	Allowed to compare any three files subject to security label restrictions. Shell-redirected I/O is subject to security label restrictions.

If the PRIV_SU configuration option is enabled, the super user is allowed to compare any three files. Shell-redirected I/O on behalf of the super user is not subject to file protections.

BUGS

Text lines that consist of a single . defeat the -e option.

There is an arbitrary limit of 200 on the total number of disagreeing ranges of text.

EXAMPLES

The following command applies the resulting script to *file1*:

```
(cat script; echo '1,$p') | ed - file1
```

FILES

TMPDIR/d3*	Temporary working file
/usr/lib/diff3prog	Executable file

SEE ALSO

diff(1)

NAME

diffmk – Marks differences between versions of a troff(1) input file

SYNOPSIS

diffmk *oldfile newfile markedfile*

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The diffmk command compares two versions of a file and creates a third version that includes change mark (.mc) commands for nroff(1) and troff(1). *oldfile* and *newfile* are the old and new versions of the file. diffmk generates *markedfile*, which contains the text from *newfile* with troff(1) change mark commands (.mc) inserted where *newfile* differs from *oldfile*. When *markedfile* is formatted, changed or inserted text is shown by a | symbol at the right margin of each line. The position of deleted text is shown by a single asterisk (*).

The diffmk command can also be used in conjunction with the proper troff(1) requests to produce program listings with marked changes, as shown in the following command line:

```
diffmk old.c new.c marked.c ; nroff reqs marked.c | pr
```

The file reqs contains the following troff(1) requests:

```
.pl 1
.ll 77
.nf
.eo
.nh
```

These troff(1) requests eliminate page breaks, adjust the line length, set no-fill mode, ignore escape characters, and turn off hyphenation, respectively.

If the characters | and * are inappropriate, you can run *markedfile* through the sed(1) command to globally change them.

NOTES

Aesthetic considerations may dictate manual adjustment of some output. File differences involving only formatting requests may produce undesirable output, that is, replacing .sp by .sp 2 produces a change mark on the preceding or following line of output.

SEE ALSO

nroff(1), sed(1), troff(1)

NAME

`dircmp` – Compares directories

SYNOPSIS

`dircmp` [`-d`] [`-s`] [`-w` *n*] *dir1 dir2*

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The `dircmp` utility examines *dir1* and *dir2* and generates various tabulated information about the contents of the directories.

Listings of files unique to each directory are generated for all options. If you do not specify an option, a list is output, indicating whether the files common to both directories have the same contents.

The `dircmp` utility accepts the following options:

`-d`	Compares the contents of files with the same name in both directories and outputs a list indicating what must be changed in the two files to bring them into agreement. The list format is described in `diff`(1).
`-s`	Suppresses messages about identical files.
`-w` *n*	Changes the width of the initial multicolumn output to *n* characters. The default width is 72.
dir1 dir2	Directories to be compared.

SEE ALSO

`cmp`(1), `diff`(1)

NAME

`domainname` – Sets or displays name of current network information service (NIS) domain

SYNOPSIS

`domainname` [*nameofdomain*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

Without an argument, `domainname` displays the name of the current domain. Only the super user can set the domain name by giving an argument; this is usually done in the `/etc/rc` start-up script. Currently, domains are used only by the NIS to refer collectively to a group of hosts.

SEE ALSO

`getdomain`(3C), `ypclnt`(3C) in the *UNICOS System Libraries Reference Manual*, Cray Research publication SR–2080

NAME

du – Summarizes disk usage

SYNOPSIS

```
du [-a] [-B | -k] [-mrRx] [file ...]
du -A [-B | -k] [-mrRx] [file ...]
du -s [-B | -k] [-mrRx] [file ...]
```

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4
CRI extensions (-A, -B, -m, and -R options)

DESCRIPTION

The du utility, by default, gives the size (in 512-byte units) of each *file* operand and (recursively) each subdirectory of the specified *file* operand. To change the default unit size, use the -B or -k options. If you omit *file* operands, the current directory is used.

The du utility accepts the following options:

-a In addition to the default output, reports the size of each nondirectory in the file hierarchy. Regardless of the presence of the -a option, nondirectories given as *file* operands are always reported.

-A Generates an entry for each file but not for directories.

-B Expresses all block counts in terms of 4096-byte units, rather than the default 512-byte units.

-k Expresses all block counts in terms of 1024-byte units, rather than the default 512-byte units.

-m Reports block counts for migrated (offline) files and nonmigrated (online) files. Valid only with Cray Research file systems. See the EXAMPLES section for the format of this output.

-r Generates messages about directories that cannot be read and files that cannot be opened. If one of these errors occurs, the exit status of the du utility will be nonzero. (This option is on by default.)

-R Suppresses messages about directories that cannot be read and files that cannot be opened, and exit status is not affected if one of these errors occurs.

-s Reports only the grand total for each of the specified *file* operands.

-x Reports only the size of files that have the same device (st_dev field in the stat structure) as the file specified by the *file* operand.

NOTES

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
`system`, `secadm`	Allowed access to any file.
	(All other users are allowed access to any file, subject to security label restrictions. Shell-redirected I/O is subject to security label restrictions.)

If the `PRIV_SU` configuration option is enabled, the super user is allowed access to any file.

The default unit size for reporting the number of blocks was changed to 512 bytes to accommodate the POSIX 1003.2 standard. When file space is computed, the block size for the file system (`f_bsize` field in the `statfs` structure) is used to compute the number of 512-byte blocks. This means that NFS file systems will report numbers consistent with the actual size of the file.

Regardless of the presence of `-a` or `-A` options, nondirectories given as *file* operands are always listed.

A file with two or more links is counted only once.

If more than 1000 distinct linked files exist, the `du` utility counts the excess files more than once.

EXIT STATUS

The `du` utility exits with one of the following values:

0 Successful completion.

>0 An error occurred.

EXAMPLES

Example 1: The following example lists disk usage by directory for directories found in `/b`. The list is sorted by descending usage:

```
$ du -s /b/* | sort -k 0nr
```

Example 2: The following example lists disk usage for online and offline files in the current directory:

```
$ du -am
4          online   0          offline file1
4          online   0          offline .
$
```

SEE ALSO

`df`(1) for information about free disk space on file systems

`stat`(2),
`statfs`(2) for information for file system statistics
in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

General UNICOS System Administration, Cray Research publication SG–2301

NAME

echo – Echoes arguments

SYNOPSIS

echo [*args*]

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The echo utility writes its arguments, separated by <blank>s and terminated by a <newline>, on standard output. It also recognizes C-like escape conventions; beware of conflicts with the shell's use of \ as in the following:

\a Writes an <alert> character.

\b Writes a <backspace> character.

\c Prints line without a <newline> (terminates the echo string).

\f Writes a <form-feed> character.

\n Writes a <newline> character.

\r Writes a <carriage-return> character.

\t Writes a <tab> character.

\v Writes a <vertical-tab> character.

\\ Writes a backslash character.

\0*n* Writes an 8-bit value that is the zero-, one-, two- or three-digit octal number *n*.

The echo utility is useful for producing diagnostics in command files and for sending known data into a pipe.

NOTES

When representing an 8-bit character by using the escape convention \0*n*, the *n* must always be preceded by the digit 0. For example, typing the following: echo 'WARNING:\07' prints the phrase WARNING: and sounds the ''bell'' on your terminal. The use of single (or double) quotation marks (or two backslashes) is required to protect the ''\'' that precedes the 07.

Following the \0, up to three digits are used in constructing the octal output character. If, following the \0*n*, you want to echo additional digits that are not part of the octal representation, you must use the full 3-digit *n*. For example, if you want to echo ESC 7, you must use the three digits 033 rather than just the two digits 33 after the \0.

For the octal equivalents of each character, see the ascii(7) man page.

csh(1) has a built-in echo utility with slightly different characteristics. See the csh(1) man page.

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
system, secadm	In a privileged administrator shell environment, allowed to write shell-redirected output to any file.
sysadm	Shell-redirected output is subject to security label restrictions.

If the PRIV_SU configuration option is enabled, the super user can write shell-redirected output to any file.

EXIT STATUS

The echo utility exits with one of the following values:

0 Successful completion.

>0 An error occurred.

EXAMPLES

Example 1: This example could be in a standard shell script prompting a user for the name:

```
echo "Enter name: \c"
read name
```

Example 2: This example prints the error message shown in double quotation marks to file test:

```
$ echo "Usage: command [-a] [-b] file" >test
```

SEE ALSO

csh(1), sh(1)

ascii(7) (available only online)

NAME

ed, red – Text editor

SYNOPSIS

ed [-p *string*] [-s] [-x] [-C] [*file*]
red [-p *string*] [-s] [-x] [-C] [*file*]

Obsolescent version; may not be supported in future releases:

ed [-p *string*] [-x] [-C] [-] [*file*]
red [-p *string*] [-x] [-C] [-] [*file*]

IMPLEMENTATION

Cray PVP systems
CRAY T3D systems

STANDARDS

POSIX, XPG4
AT&T extensions (-x and -C options)

DESCRIPTION

The ed utility is the standard text editor. If you specify file, ed simulates an e command (see the following text) on the specified file; that is, the file is read into the ed buffer so that you can edit it.

The ed utility accepts the following options:

-p *string* Lets users specify a prompt string.

-s Suppresses the printing of character counts by e, E, r, and w commands, of diagnostics from e and q commands, and of the ! prompt after a !shell command. See the WARNINGS section.

-x Encryption option; when used, ed simulates an X command and prompts the user for a key, which is used to encrypt and decrypt text using the algorithm of crypt(1). The X command makes an educated guess to determine whether or not text read in is encrypted. The temporary buffer file is encrypted also, using a transformed version of the key typed in for the -x option. See crypt(1) and the WARNINGS section.

-C Encryption option; the same as the -x option, except that ed simulates a C command. The C command is like the X command, except that all text read in is assumed to have been encrypted. See the WARNINGS section.

- (Obsolescent) Equivalent to -s.

The `ed` utility operates on a copy of the file it is editing; changes made to the copy have no effect on the file until you specify a `w` (write). The copy of the text being edited resides in a temporary file called the *buffer*. Only one buffer exists.

The `red` utility is a restricted version of `ed`. It allows the editing of files only in the current directory. It prohibits the execution of shell commands through `!shell` *command*. If you try to bypass these restrictions, an error message (`restricted shell`) results.

Commands to `ed` have a simple and regular structure: zero, one, or two addresses followed by a single-character command, possibly followed by arguments to that command. These addresses specify one or more lines in the buffer. Every command that requires addresses has default addresses; therefore, you can usually omit the addresses.

Generally, only one command can appear on a line. Certain commands allow the input of text, which is placed in the appropriate place in the buffer. When `ed` is accepting text, it is in input mode. In this mode, commands are not recognized; all input is merely collected. To leave input mode, type a single period (`.`) at the beginning of a line, and immediately press `<RETURN>`.

The `ed` utility supports a limited form of regular expression notation; regular expressions are used in addresses to specify lines and, in some commands (for example, `s`), to specify portions of a line that will be substituted. A regular expression (RE) specifies a set of character strings. A member of this set of strings is said to be *matched* by the RE. The REs allowed by `ed` are constructed as follows:

The following 1-character REs match a single character:

1. An ordinary character (not one of those discussed in item 2) is a 1-character RE that matches itself.
2. A backslash (`\`) followed by any special character is a 1-character RE that matches the special character itself. The special characters are as follows:
   a. A period (`.`), asterisk (`*`), left bracket, (`[`), or backslash (`\`), which are always special, except when they appear within brackets (`[]`; see item 4).
   b. A caret or circumflex (`^`), which is special at the beginning of an entire RE (see items 11 and 13), or when it immediately follows the left of a pair of brackets (`[]`) (see item 4).
   c. A dollar sign (`$`), which is special at the end of an entire RE (see item 12).
   d. The character used to bound (such as, delimit) an entire RE, which is special for that RE (for example, see how slash (`/`) is used in the *g* command).
3. A period (`.`) is a 1-character RE that matches any character except a `<newline>`.

4. A nonempty string of characters enclosed in brackets (`[]`) is a 1-character RE that matches any one character in that string. If, however, the first character of the string is a circumflex (`^`), the 1-character RE matches any character except `<newline>` and the remaining characters in the string. The `^` has this special meaning only if it occurs first in the string. You can use the minus sign (`-`) to indicate a range of consecutive ASCII characters (for example, `[0-9]` is equivalent to `[0123456789]`). The `-` loses this special meaning if it occurs first (after an initial `^`, if any) or last in the string. The right bracket (`]`) does not terminate such a string when it is the first character within it (after an initial `^`, if any) (for example, `[]a-f]` matches either a right bracket (`]`) or one of the letters `a` through `f`, inclusive). The four characters listed in item 2 stand for themselves within such a string of characters.

You can use the following rules to construct REs from 1-character REs:

5. A 1-character RE is a RE that matches whatever the 1-character RE matches.
6. A 1-character RE followed by an asterisk (`*`) is a RE that matches zero or more occurrences of the 1-character RE. If there is any choice, the longest leftmost string that permits a match is chosen.
7. A 1-character RE followed by `\{`*m*`\}`, `\{`*m*`,\}`, or `\{`*m*`,`*n*`\}` is a RE that matches a range of occurrences of the 1-character RE. The values of *m* and *n* must be nonnegative integers less than 256; `\{`*m*`\}` matches exactly *m* occurrences; `\{`*m*`,\}` matches at least *m* occurrences; `\{`*m*`,`*n*`\}` matches any number of occurrences between *m* and *n*, inclusive. Whenever a choice exists, the RE matches as many occurrences as possible.
8. The concatenation of REs results in an RE that matches the concatenation of the strings matched by each component of the RE.
9. A RE enclosed between the character sequences `\(` and `\)` is a RE that matches whatever the unadorned RE matches.
10. The expression `\`*n* matches the same string of characters as was matched by an expression enclosed between `\(` and `\)` earlier in the same RE. Here, *n* is a digit; the subexpression specified is that beginning with the *n*th occurrence of `\(` counting from the left. For example, the expression `^\(.*\)\1$` matches a line that consists of two repeated appearances of the same string.

Finally, an entire RE may be constrained to match only an initial or final segment of a line (or both).

11. A circumflex (`^`) at the beginning of an entire RE constrains that RE to match an initial segment of a line.
12. A dollar sign (`$`) at the end of an entire RE constrains that RE to match a final segment of a line.
13. The construction `^`*entire RE*`$` constrains the entire RE to match the entire line.

The null RE (such as `//`) is equivalent to the last RE encountered.

To understand addressing in `ed`, you must know that at any time there is a current line. Generally, the current line is the last line affected by a command; the exact effect on the current line is discussed under the description of each command. Addresses are constructed as follows:

1. The character `.` addresses the current line.
2. The character `$` addresses the last line of the buffer.

3. A decimal number *n* addresses the *n*th line of the buffer.
4. ′*x* addresses the line marked with the mark name character *x*, which must be an ASCII lowercase letter (`a` – `z`). Lines are marked with the `k` command described later in this man page.
5. A RE enclosed by slashes (`/`) addresses the first line found by searching forward from the line following the current line toward the end of the buffer and stopping at the first line containing a string matching the RE. If necessary, the search wraps around to the beginning of the buffer and continues up to and including the current line, so that the entire buffer is searched.
6. An RE enclosed in question marks (`?`) addresses the first line found by searching backward from the line preceding the current line toward the beginning of the buffer and stopping at the first line containing a string matching the RE. If necessary, the search wraps around to the end of the buffer and continues up to and including the current line.
7. An address followed by a plus sign (`+`) or a minus sign (`-`) followed by a decimal number specifies that address plus or minus the indicated number of lines. You can omit the plus sign.
8. If an address begins with `+` or `-`, the addition or subtraction is taken with respect to the current line (for example, `-5` means `.-5`).
9. If an address ends with `+` or `-`, 1 is added to or subtracted from the address, respectively. As a consequence of this rule and of rule 8, in this list, the address `-` refers to the line preceding the current line. (To maintain compatibility with earlier versions of the editor, the `^` character in addresses is entirely equivalent to `-`.) Moreover, trailing `+` and `-` characters have a cumulative effect, so `- -` refers to the current line less 2.
10. For convenience, a comma (`,`) stands for the address pair `1,$`, and a semicolon (`;`) stands for the pair `.,$`.

Commands may require zero, one, or two addresses. Commands that require no addresses regard the presence of an address as an error. Commands that accept one or two addresses assume default addresses when an insufficient number of addresses is given; if more addresses are given than such a command requires, the last one(s) are used.

Typically, addresses are separated from each other by a comma (`,`). They may also be separated by a semicolon (`;`). In the latter case, the current line (`.`) is set to the first address, and only then is the second address calculated. This feature can be used to determine the starting line for forward and backward searches (see rules 5 and 6, in the preceding list). The second address of any two-address sequence must correspond to a line that follows, in the buffer, the line corresponding to the first address.

In the following list of `ed` commands, the default addresses are shown in parentheses. The parentheses are not part of the address; they show that the specified addresses are the default.

It is generally illegal for more than one command to appear on a line. However, any command (except `e`, `f`, `r`, or `w`) may be suffixed by `l`, `n`, or `p`; in which case, the current line is listed, numbered, or printed, respectively, as discussed under the `l`, `n`, and `p` commands in the following list:

`(.)a`
<*text*>

`.` The command reads the text entered and appends it after the addressed line; `.` is left at the last inserted line, or, if there were none, at the addressed line. Address 0 is legal for this command: it causes the appended text to be placed at the beginning of the buffer. The maximum number of characters that may be entered from a terminal is 256 per line (including the `<newline>` character).

`(.)b`*a* The browse command displays the addressed line and *a* lines below it; *a* is a number. The initial default for *a* is 22 lines; when you give a different value, that value becomes the default. `.` is left at the last line displayed. You may append the `b` command to any other command except `e`, `f`, `r`, or `w`.

`(.)c`
<text>
`.` The change command deletes the addressed lines, then accepts input text that replaces these lines; `.` is left at the last line input, or, if there were none, at the first line that was not deleted.

`(.,.)d` The delete command deletes the addressed lines from the buffer. The line after the last line deleted becomes the current line; if the lines deleted were originally at the end of the buffer, the new last line becomes the current line.

`e` *file* The edit command deletes the entire contents of the buffer, and then reads in the specified file; `.` is set to the last line of the buffer. If no file name is given, the currently remembered file name, if any, is used (see the `f` command). The number of characters read is typed; *file* is remembered for possible use as a default file name in subsequent `e`, `r`, and `w` commands. If *file* is replaced by `!`, the rest of the line is taken to be a shell (`sh`(1)) command whose output is to be read. Such a shell command is not remembered as the current file name. See also the MESSAGES section.

`E` *file* The edit command is like `e`, except that the editor does not check to see whether any changes were made to the buffer since the last `w` command.

`f` *file* If you specify *file*, the file-name command changes the currently remembered file name to *file*; otherwise, it prints the currently remembered file name.

`(1,$)g/`*RE*`/`*command list*
In the global command, the first step is to mark every line that matches the specified RE. Then, for every such line, the given *command list* is executed with `.` initially set to that line. A single command or the first of a list of commands appears on the same line as the global command. You must end all lines of a multiline list, except the last line with a `\`; `a`, `i`, and `c` commands and associated input are permitted. You can omit the `.` terminating input mode if it would be the last line of the *command list*. An empty *command list* is equivalent to the `p` command. The `g`, `G`, `v`, and `V` commands are not permitted in the *command list*. See also the BUGS section.

`(1,$)G/`*RE*`/` In the interactive global command, the first step is to mark every line that matches the given RE. Then, for every such line, that line is printed, `.` is changed to that line, and any one command (other than one of the `a`, `c`, `i`, `g`, `G`, `v`, and `V` commands) may be input and is executed. After the execution of that command, the next marked line is printed, and so on; a `<newline>` acts as a null command; `&` causes the reexecution of the most recent command executed within the current invocation of `G`. The commands input as part of the execution of the `G` command may address and affect any lines in the buffer. You can terminate the `G` command with an interrupt signal (ASCII DEL or BREAK).

`h` The `h`elp command gives a short error message that explains the reason for the most recent `?` diagnostic message.

`H` The `H`elp command causes `ed` to enter a mode in which error messages are printed for all subsequent `?` diagnostic messages. It also explains the previous `?` if one exists. The `H` command alternately turns this mode on and off; it is initially off.

`help` The `help` command gives a one-page synopsis of `ed` commands.

`(.)i`
<text>
`.` The `i` command inserts the specified text before the addressed line; `.` is left at the last inserted line, or, if there were none, at the addressed line. This command differs from the `a` command only in the placement of the input text. Address 0 is not legal for this command. The maximum number of characters that may be entered from a terminal is 256 per line (including the `<newline>` character).

`(.,.+1)j` The join command joins contiguous lines by removing the appropriate `<newline>` characters. If exactly one address is given, this command does nothing.

`(.)k` *x* The mark command marks the addressed line with name *x*, which must be a lowercase letter. The address *'x* then addresses this line; `.` is unchanged.

`(.,.)l` The list command prints the addressed lines in an unambiguous way: a few nonprinting characters (for example, `<tab>` and `<backspace>`) are represented by visually-mnemonic overstrikes. All other nonprinting characters are printed in octal, and long lines are folded. You may append an `l` command to any command other than `e`, `f`, `r`, or `w`.

`(.,.)m` *a* The move command repositions the addressed line(s) after the line addressed by `a`. Address 0 is legal for `a` and causes the addressed line(s) to be moved to the beginning of the file. Address `a` must not fall within the range of moved lines; `.` is left at the last line moved.

`(.,.)n` The number command prints the addressed lines, preceding each line by its line number and a tab character; `.` is left at the last line printed. You can append the `n` command to any other command other than `e`, `f`, `r`, or `w`.

`N` The `N` command toggles the functions of the `p` and `n` commands. Entering `N` causes `n` to function as `p` and vice versa. Another `N` toggles them back.

`(.)o`*a* The `o` command displays the addressed line and *a* lines above and below it; *a* is a number. The initial default for *a* is 11 lines; when you specify a different value, that value becomes the default. `.` is left at the line addressed by `o`. You can append the `o` command to any other command except `e`, `f`, `r`, or `w`.

`(.,.)p` The print command prints the addressed lines; `.` is left at the last line printed. You can append the `p` command to any other command other than `e`, `f`, `r`, or `w`. For example, `dp` deletes the current line and prints the new current line.

`P` The editor prompts with a `*` for all subsequent commands. The `P` command alternately turns this mode on and off; it is initially off.

`q` The quit command causes `ed` to exit. No automatic write of a file is done. See also the MESSAGES section.

`Q` The editor exits without checking whether changes have been made in the buffer since the last `w` command.

`($)r` *file* The read command reads in the given file after the addressed line. If you omit *file*, the currently remembered file name, if any, is used (see commands `e` and `f`). The currently-remembered file name is not changed unless *file* is the very first file name mentioned since `ed` was invoked. Address 0 is legal for `r` and causes the file to be read at the beginning of the buffer. If the read is successful, the number of characters read is typed; `.` is set to the last line read in. If *file* is replaced by `!`, the rest of the line is taken to be a shell (`sh`(1)) command whose output will be read. For example, `$r !ls` appends the file names in the current directory to the end of the file being edited. Such a shell command is not remembered as the current file name.

`(.,.)s/`*RE*`/`*replacement*`/` or
`(.,.)s/`*RE*`/`*replacement*`/g` or

(.,.)s/*RE/replacement*/n *n* = 1–512

The substitute command searches each addressed line for an occurrence of the specified RE. In each line in which a match is found, all (nonoverlapped) matched strings are replaced by the *replacement* if global replacement indicator g appears after the command. If the global indicator does not appear, only the first occurrence of the matched string is replaced. If a number *n* appears after the command, only the *n*th occurrence of the matched string on each addressed line is replaced. It is an error for the substitution to fail on all addressed lines. Any character other than <space> or <newline> may be used instead of / to delimit the RE and the *replacement*; . is left at the last line on which a substitution occurred.

An ampersand (&) that appears in the *replacement* is replaced by the string matching the RE on the current line. To suppress the special meaning of & in this context, precede it by \. As a more general feature, the characters *n*, where *n* is a digit, are replaced by the text matched by the nth regular subexpression of the specified RE enclosed between \(and \). When nested parenthesized subexpressions are present, *n* is determined by the number of occurrences of \(starting from the left. When the character % is the only character in the *replacement*, the *replacement* used in the most recent substitute command is used as the *replacement* in the current substitute command. The % loses its special meaning when it is in a replacement string of more than one character or is preceded by a \.

You may split a line by substituting a <newline> character into it. The <newline> in the *replacement* must be escaped by preceding it with \. Such substitution cannot be done as part of a g or v command list.

(.,.)t*a* This command acts just like the m command, except that a copy of the addressed lines is placed after address a (which may be 0); . is left at the last line of the copy.

u The undo command nullifies the effect of the most recent command that modified anything in the buffer, namely the most recent a, c, d, g, i, j, m, r, s, t, v, G, or V command.

(1,$)v/*RE/command list*

This command is the same as the global command g except that the *command list* is executed with . initially set to every line that does not match the RE.

(1,$)V/*RE/* This command is the same as the interactive global command G except that the lines that are marked during the first step are those that do not match the RE.

`(1,$)w` *file* The write command writes the addressed lines into the specified file. If the file does not exist, it is created with mode 666 (readable and writable by everyone), unless your `umask` setting (see `umask`(1)) dictates otherwise. The currently remembered file name is not changed unless *file* is the very first file name mentioned since `ed` was invoked. If you omit *file*, the currently remembered file name, if any, is used (see commands `e` and `f`); `.` is unchanged. If the command is successful, the number of characters written is typed. If *file* is replaced by `!`, the rest of the line is taken to be a shell (`sh`(1)) command whose standard input is the addressed lines. Such a shell command is not remembered as the current file name.

`X` An encryption key is requested from the standard input. Subsequent `e`, `r`, and `w` commands use this key to encrypt or decrypt the text (see `crypt`(1)). An explicitly empty key turns off encryption. Also, see the `-x` option of `ed`.

`z` The `z` command is a UNICOS extension that writes the file under the original name and then exits `ed`. It is functionally equivalent to entering the `w` and `q` commands.

`($)=` The line number of the addressed line is typed; `.` is unchanged by this command.

`!`*shell command* The remainder of the line after the `!` is sent to the UNIX system shell (`sh`(1)) to be interpreted as a command. Within the text of that command, unescaped character `%` is replaced with the remembered file name; if a `!` appears as the first character of the shell command, it is replaced with the text of the previous shell command. Thus, `!!` repeats the last shell command. If any expansion is performed, the expanded line is echoed; `.` is unchanged.

`(.+1) <newline>`
An address alone on a line causes the addressed line to be printed. A `<newline>` alone is equivalent to `.+1p`; it is useful for stepping forward through the buffer.

If an interrupt signal (ASCII DEL or BREAK) is sent, `ed` prints `?` and returns to its command level.

Some size limitations: 512 characters per line, 256 characters per global command list, and 64 characters per file name. The limit on the number of lines depends on the amount of user memory: each line takes 1 word.

When reading a file, `ed` discards ASCII null characters.

If a file is not terminated by a `<newline>` character, `ed` adds one and outputs a message that explains what it did.

If the closing delimiter of a RE or of a replacement string (such as, `/`) would be the last character before a new line, that delimiter may be omitted, in which case the addressed line is printed. The following pairs of commands are equivalent:

```
s/s1/s2   s/s1/s2/p
g/s1      g/s1/p
?s1       ?s1?
```

WARNINGS

Inclusion of the Data Encryption Standard (DES) encryption code requires a special license for sites outside the United States and Canada. If these encryption functions are not available on your system, check with your system administrator or site analyst.

The `-` option, although supported in this release for upward compatibility, will no longer be supported in the next major release of the system. Convert shell scripts that use the `-` option so that they use the `-s` option instead.

Reasonable editing sessions should be kept under 10 Mbytes. Lines are limited to 4096 characters.

When reading a file, `ed` discards ASCII null characters and all characters after the last `<newline>`. The `ed` utility cannot edit files (such as `a.out`) that contain characters that are not in the ASCII set (bit 8 on).

Size limitations: Large files generate larger editor temporary files and cost many processor cycles on entry to `ed`.

Files encrypted on Cray Research systems using a UNICOS release prior to 5.0 must be reencrypted using `crypt`(1) before `ed -x` can read them.

EXIT STATUS

The `ed` utility exits with one of the following values:

0 Successful completion without any file or command errors.

>0 An error occurred.

MESSAGES

? For command errors.

?*file* For an inaccessible file (use the `help` and `Help` commands for detailed explanations).

When changes have been made in the buffer since the last `w` command that wrote the entire buffer, `ed` warns the user if an attempt is made to destroy the `ed` buffer by using the `e` or `q` commands. It prints `?` and allows editing to continue. A second `e` or `q` command at this point takes effect. The `-s` command-line option inhibits this feature.

BUGS

The `!` command cannot be subject to a `g` or a `v` command.

You cannot use the `!` command and the `!` escape from the `e`, `r`, and `w` commands if you invoke the editor from a restricted shell (see `sh`(1)).

The sequence `\n` in a RE does not match a `<newline>` character.

If the editor input is coming from a command file (for example, `ed file < ed-cmd-file`), the editor will exit at the first failure.

FILES

`/usr/tmp`	Default directory for temporary work file.
`TMPDIR`	If this environmental variable is not null, its value is used in place of `/usr/tmp` as the directory name for the temporary work file.
`ed.hup`	Work is saved in this file if the terminal is hung up.

SEE ALSO

`chown`(1), `crypt`(1), `edit`(1), `ex`(1), `grep`(1), `sed`(1), `sh`(1), `stty`(1), `umask`(1), `vi`(1)

`regex`(3C) in the *UNICOS System Libraries Reference Manual*, Cray Research publication SR–2080

NAME

edit – Text editor (variant of ex(1))

SYNOPSIS

edit [-r] [-x] *name*

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The edit editor is a variant of the text editor ex(1) that is recommended for new or casual users who want to use a command-oriented editor.

The edit command accepts the following options:

-r Recovers file after an editor or system crash.

-x Encryption option; when this option is used, the file is encrypted as it is being written and requires an encryption key to be read (see crypt(1)). Also, see the WARNINGS section of this man page.

name Specifies the name of a file to edit.

The following introduction will help you get started with edit. If you are using a CRT terminal you may want to learn about the display editor vi(1).

To edit the contents of an existing file, enter the command edit *name* to the shell. edit makes a copy of the file, which you can then edit, and tells you how many lines and characters are in the file. To create a new file, make up a name for the file and try to run edit on it; this will create an error message.

The edit command prompts for commands by using the colon (:) character, which you see after starting the editor. If you are editing an existing file, you will have some lines in the edit buffer (its name for the copy of the file you are editing). Most commands to edit use the current line unless you specify the line to be used. If you specify print (p), and press the carriage return (as you should after all edit commands), this current line will be printed. If you delete (d) the current line, edit will print the new current line. When you start editing, edit makes the last line of the file the current line. If you delete the current line, the next line in the file becomes the current line. (Deleting the last line is a special case.)

If you start with an empty file or want to add new lines, the append (a) command can be used. After you enter this command, edit reads lines from your terminal, placing these lines after the current line, until you enter a line consisting of just a period (.). The last line you typed then becomes the current line. The command insert is like append, but places the lines you enter before, rather than after, the current line.

The edit command numbers the lines in the buffer, with the first line having number 1. If you enter the command 1, edit will type the first line. If you then enter the command delete, edit will delete the first line, line 2 will become line 1, and edit will print the current line (the new line 1). In general, the current line is always the last line affected by a command.

You can make a change to text within the current line by using the `substitute` (`s`) command. Enter `s/`*old*`/`*new*`/`, where *old* is where you insert the old characters you want replaced and *new* is where you insert the new characters to replace the old characters.

The command `file` (`f`) tells you how many lines are in the buffer you are editing, and indicates `[Modified]` if you have changed it. After modifying a file, you can put the buffer text back to replace the file by issuing a `write` (`w`) command. You can then leave the editor by issuing a `quit` (`q`) command. If you run `edit` on a file but do not change the file, it is not necessary to `write` the file back. If you try to `quit` from `edit` after modifying the buffer without writing it out, you will be warned that there has been `No write since last change` and `edit` will wait for another command. If you wish not to `write` the buffer out, then you can issue another `quit` command. The buffer is then irretrievably discarded, and you return to the shell.

By using the `delete` and `append` commands and specifying line numbers to see lines in the file, you can make any changes. You should learn at least a few more things, however, if you are going to use `edit` very often.

The `change` (`c`) command changes the current line to a sequence of lines you supply (as in `append`, you supply lines up to a line consisting of only a period (`.`). You can use `change` to change more than one line by giving the line numbers of the lines you want to change, for example, `3,5change`. You can print lines this way too. The command `1,23p` prints the first 23 lines of the file.

The `undo` (`u`) command reverses the effect of the last command that changed the buffer. If you issue a `substitute` command that does not do what you want, you can enter `undo` to restore the old contents of the line. You can also undo an `undo` command. `edit` gives you a warning message when commands you issue affect more than one line of the buffer. If the amount of change seems unreasonable, you should consider issuing an `undo` and looking to see what happened. If you decide that the change was correct, you can `undo` again to get it back. Commands such as `write` and `quit` cannot be undone.

To look at the next line in the buffer, press the carriage return key. To look at a number of lines, press `<CONTROL-d>` (hold down the `CONTROL` key and the `d` key at the same time) rather than carriage return. This shows you a half-screen of lines on a CRT or 12 lines on a hardcopy terminal. You can look at the text surrounding the line on which you are working by issuing the command `z`. The current line is then the last line printed; you can get back to the line where you were before the `z` command by entering ´´. The `z` command can also be given other following characters: `z-` prints a screen of text (or 24 lines) ending where you are; `z+` prints the next screenful. If you want less than a screenful of lines, type in `z.12` to get 12 lines total. You can delete five lines, starting with the current line, with the command `delete 5` .

To find things in the file, you can use line numbers. Note that the line numbers change when you insert and delete lines. You can search backward and forward in the file for strings by giving commands of the form `/text/` to search forward, for *text* or `?text?` to search backward. If a search reaches the end of the file without finding the text, it wraps around, and continues to search until it reaches the line from which you entered the command. A useful feature here is a search of the form `/^text/`, which searches for text at the beginning of a line. Similarly `/text$/` searches for text at the end of a line. You can omit the trailing / or ? in these commands.

The current line has a symbolic name, a period (`.`). This is most useful in a range of lines as in the `.,$print` command, which prints the rest of the lines in the file. To get to the last line in the file, you can refer to it by its symbolic name, `$`. The command `$ delete` or `$d` deletes the last line in the file, no matter which was the current line. Arithmetic with line references is also possible. The command `$-5` takes you to the fifth line before the last line, and `.+20` moves to 20 lines after the present line.

You can determine the current line by entering `.=`. This is useful if you want to move or copy a section of text within a file or between files. Find out the first and last line numbers you want to copy or move (for example, 10 to 20). To move lines, you can enter `10,20delete a`, which deletes these lines from the file and places them in a buffer named `a`. The `edit` command has 26 buffers named a through z. You can get these lines back by entering `put a`, which will put the contents of buffer `a` after the current line. If you want to move or copy these lines between files, copy the lines, and enter an `edit` (`e`) command, following it with the name of the other file you want to edit, for example, `edit chapter2`. You can also use the `yank` command instead of `delete` to copy or move lines. If the text you want to copy or move is within one file, you can enter `10,20move $`. It is not necessary to use named buffers in this case.

WARNINGS

Inclusion of the Data Encryption Standard (DES) encryption code requires a special license for sites outside the United States and Canada. If these encryption functions are not available on your system, check with your system administrator or site analyst.

SEE ALSO

`crypt`(1), `ed`(1), `ex`(1), `vi`(1)

NAME

emacs – GNU project Emacs

SYNOPSIS

emacs [*file* ...]

IMPLEMENTATION

All Cray Research systems

STANDARDS

FSF

DESCRIPTION

GNU Emacs is written by Richard Stallman, the author of the original (PDP-10) Emacs. The primary documentation of GNU Emacs is in the *GNU Emacs Manual*, which you can read online using Info, a subsystem of Emacs. Please look there for complete and up-to-date documentation.

The user functionality of GNU Emacs encompasses everything other Emacs editors do, and it is easily extensible because its editing commands are written in Lisp.

emacs has an extensive interactive help facility, but the facility assumes that you know how to manipulate Emacs windows and buffers. <CONTROL-h> (backspace or <CONTROL-h>) enters the help facility. Help Tutorial (<CONTROL-h t>) requests an interactive tutorial. This tutorial can teach beginners the fundamentals of emacs in a few minutes. Help Apropos (<CONTROL-h a>) helps you find a command given its functionality. Help Character (<CONTROL-h c>) describes a given character's effect. Help Function (<CONTROL-h f>) describes a given Lisp function specified by name.

The emacs Undo command can undo several steps of modification to your buffers; therefore you can easily recover from editing mistakes.

The GNU Emacs contains many special packages that handle mail reading (RMail) and sending (Mail), outline editing (Outline), compiling (Compile), running subshells within Emacs windows (Shell), and running a Lisp read-eval-print loop (Lisp-Interaction-Mode).

An extensive reference manual exists, but users of other versions of Emacs should have little trouble adapting even without a copy. Users new to Emacs use basic features fairly rapidly by studying the tutorial and using the self-documentation features.

emacs accepts the following options:

file Edits *file*.

+*number* Goes to the line specified by *number* (does not insert a space between the "+" sign and the number).

`-d` *displayname*
Creates the Emacs window on the display specified by *displayname.* This must be the first argument listed in the command line.

`-q` Does not load an init file.

`-u` *user* Load *user's* init file.

`-t` *file* Use specified *file* as the terminal, rather than using `stdin`/`stdout`. This must be the first argument specified in the command line.

The following options are Lisp-oriented (these options are processed in the order encountered):

`-f` *function* Execute the Lisp function *function*.

`-l` *file* Load the Lisp code in file *file*.

The following options are useful when running `emacs` as a batch editor:

`-batch` *commandfile*
Edit in batch mode by using the commands found in *commandfile*. The editor sends messages to `stdout`. This option must be the first one in the argument list.

`-kill` Exit `emacs` while in batch mode.

Using `emacs` with X Windows

`emacs` has been tailored to work well with the X Window System environment. If you run `emacs` using the X Window System, it will create its own display window. You should start the editor as a background process so that you can continue using your original window. `emacs` can be started with the following X Window System switches:

`-r` Displays the Emacs window in reverse video.

`-i` Uses the "gnu" bit map icon when iconifying the Emacs window.

`-font` *font* Sets the Emacs window's font to that specified by *font*. Emacs accepts only fixed-width fonts. Under the X11 Release 4 font-naming conventions, any font with the value `m` or `c` in the eleventh field of the font name is a fixed-width font. Furthermore, fonts whose name are of the form 'width x height' are generally fixed-width, as is the font `fixed`.

When you specify a font, do not include the *.onx* extension; be sure to put a space between the `-font` switch and the font specification argument.

`-b` *pixels* Sets the Emacs window's border width to the number of pixels specified by *pixels*. The defaults is one pixel on each side of the window.

`-ib` *pixels* Sets the Emacs window's internal border width to the number of pixels specified by *pixels*. The defaults is one pixel of padding on each side of the window.

`-geometry` *geometry*
Sets the Emacs window's width, height, and position as specified. The geometry specification is in the standard X format. The width and height are specified in characters; the default is 80 by 24.

`-fg` *color* On color displays, sets the color of the text to *color*.

`-bg` *color* On color displays, sets the color of the window's background to *color*.

`-bd` *color* On color displays, sets the color of the window's border to *color*.

`-cr` *color* On color displays, sets the color of the window's text cursor to *color*.

`-ms` *color* On color displays, sets the color of the window's mouse cursor.

`-d` *displayname*
Creates the Emacs window on the display specified by *displayname*. This option must be the first one specified in the command line.

`-nw` Disables the Emacs special interface to the X Window System environment. If you use this switch when invoking Emacs from an *xterm* window, display is done in the *xterm* window. This option must be the first one specified in the command line.

You can set *X* default values for your Emacs windows in your *.Xdefaults* file. Use the following format:

`emacs.`*keyword*`:`*value*

where *value* specifies the default value of *keyword*.

Emacs lets you set default values for the following keywords:

`BodyFont` Sets the window's text font.

`ReverseVideo` If the value of `ReverseVideo` is set to `on`, the window will be displayed in inverse video.

`BitMapIcon` If the value of `BitMapIcon` is set to `on`, the window will iconify using the "gnu" icon.

`BorderWidth` Sets the window's border width in pixels.

`Foreground` For color displays, sets the window's text color.

`Background` For color displays, sets the window's background color.

`Border` For color displays, sets the color of the window's border.

`Cursor` For color displays, sets the color of the window's text cursor.

`Mouse` For color displays, sets the color of the window's mouse cursor.

If you try to set color values when using a black and white display, the window's characteristics will default as follows: the foreground color will be set to black, the background color will be set to white, the border color will be set to gray, and the text and mouse cursors will be set to black.

Using the Mouse

The following table lists the key bindings for the mouse cursor when used in an Emacs window.

	Left Button	Middle Button	Right Button
Unshifted	Select window set point	Paste text	Cut text
Shift		Cut text	Paste text
Control		Cut & wipe text	Select & split into two wndows
Control + Shift	X buffer menu †	X help menu ‡	Keep one window

† X buffer menu: hold the buttons and keys down, wait for menu to appear, select buffer, and release. Move mouse out of menu and release to cancel.

‡ X help menu: pop up index card menu for Emacs help.

Manuals

You can order printed copies of the *GNU Emacs Manual* from the Free Software Foundation, which develops GNU software. See the file `ORDERS` for ordering information.

NOTES

Emacs is free; anyone may redistribute copies of Emacs to anyone under the terms stated in the Emacs General Public License, a copy of which accompanies each copy of Emacs and which also appears in the reference manual.

Copies of Emacs may sometimes be received packaged with distributions of UNIX systems, but it is never included in the scope of any license covering those systems. Such inclusion violates the terms on which distribution is permitted. In fact, the primary purpose of the General Public License is to prohibit anyone from attaching any other restrictions to redistribution of Emacs.

Richard Stallman encourages you to improve and extend Emacs. You contribute your extensions to the GNU library. Eventually GNU (GNU's Not UNIX) will be a complete replacement for Berkeley UNIX. Everyone will be able to use the GNU system for free.

AUTHORS

Emacs was written by Richard Stallman and the Free Software Foundation. Joachim Martillo and Robert Krawitz added the X Window System features.

FILES

`/usr/src/prod/emacs/src`
: C source files and object files.

`/usr/lib/emacs/lisp`
: Lisp source files and compiled files that define most editing commands. Some are preloaded; others are autoloaded from this directory when used.

`/usr/src/prod/emacs/man`
Sources for the Emacs Reference Manual.

`/usr/lib/emacs/etc`
Various programs that are used with GNU Emacs, and some files of information.

`/usr/lib/emacs/etc/DOC.*`
Contains the documentation strings for the Lisp primitives and preloaded Lisp functions of GNU Emacs. They are stored here to reduce the size of Emacs proper.

`/usr/lib/emacs/etc/DIFF`
Discusses GNU Emacs versus Twenex Emacs.

`/usr/lib/emacs/etc/CCADIFF`
Discusses GNU Emacs versus CCA Emacs.

`/usr/lib/emacs/etc/GOSDIFF`
Discusses GNU Emacs versus Gosling Emacs.

`/usr/lib/emacs/info`
Identifies files to which the Info documentation browser (a subsystem of Emacs) may refer. Currently not much of UNIX is documented here, but the complete text of the Emacs reference manual is included in a convenient tree-structured form.

BUGS

The shell will not work with programs running in raw mode on some version of UNIX software.

There is a mailing list, `bug-gnu-emacs@prep.ai.mit.edu` on the internet (`ucbvax!prep.ai.mit.edu!bug-gnu-emacs` on UUCPnet), for reporting Emacs bugs and fixes. But before reporting something as a bug, try to be sure that it really is a bug, and not a misunderstanding or a deliberate feature. Read the section ''Reporting Emacs Bugs'' near the end of the Emacs Reference Manual (or Info system) for hints on how and when to report bugs. Also, include the version number of the Emacs you are running in *every* bug report that you submit.

Do not expect a personal answer to a bug report. Reporting bugs gets them fixed for everyone in the next release, if possible.

Do not send anything but bug reports to this mailing list. Send requests to be added to mailing lists to the special list `info-gnu-emacs-request@prep.ai.mit.edu` (or the corresponding UUCP address). For more information about Emacs mailing lists, see the file `/usr/lib/emacs/etc/MAILINGLISTS`. Bugs tend to be fixed if they can be isolated, so it is in your interest to report them in such a way that they can be reproduced easily.

NAME

env – Sets environment for command execution

SYNOPSIS

env [-i] [*env-vars*]... [*utility* [*args*]]

Obsolescent version; may not be supported in future releases:

env [-] [*env-vars*]... [*utility* [*args*]]

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The env utility obtains the current environment, modifies it according to its arguments, and then executes the *utility* and its *args*, if any, with the modified environment.

You can specify zero or more *env-vars* arguments that must be of the form *name=value*. These arguments are merged into the inherited environment before *utility* is executed.

The env utility accepts the following options:

-

-i Causes the inherited environment to be ignored completely, so that *utility* is executed with the environment specified by the arguments.

If you omit *utility*, the resulting environment is printed, one name-value pair per line.

CAUTIONS

The env utility allows a maximum of 255 environment variable names.

STATUS EXIT

If the *utility* utility is invoked, the exit status of env is the exit status of *utility*; otherwise, the env utility exits with one of the following values:

0 The env utility successfully completed.

1–125 An error occurred in the env utility.

126 The utility specified by *utility* was found but could not be invoked.

127 The utility specified by *utility* could not be found.

EXAMPLES

The following example performs a profile run of `mycode.f` source file and alters the size of the statistical buckets only for the duration of the run. The file is compiled by using `f90`(1). `segldr`(1) then loads with the `prof` library. Finally, `env` sets the value of the `PROF_WPB` environment variable to 1 only for the duration of this
execution.

```
f90 -g mycode.f
segldr -l prof mycode.o
env PROF_WPB=1 a.out
```

If the `PROF_WPB` environment variable was set previously to some value in your global environment variables, the `env` execution will temporarily override the value.

SEE ALSO

`sh`(1),

`exec`(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

`profile`(5) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

NAME

eqn, neqn, checkeq – Typesets mathematics

SYNOPSIS

```
eqn [-dxy] [-fn] [-pn] [-sn] [-Tdev] [filename] ...
neqn [filename] ...
checkeq [filename] ...
```

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The eqn and neqn commands are language processors to assist in describing equations. eqn is a preprocessor for troff(1) and is intended for devices that can print output from troff. The neqn command is a preprocessor for nroff(1) and is intended for use with terminals.

The checkeq command reports missing or unbalanced delimiters and .EQ/.EN pairs.

If no *filename*s are specified, eqn and neqn read from the standard input. A line beginning with .EQ marks the start of an equation; the end of an equation is marked by a line beginning with .EN. Neither of these lines is altered, so they may be defined in macro packages to get centering, numbering, and so on. It is also possible to set 2 characters as delimiters; subsequent text between delimiters is also treated as eqn input.

The following options are available for eqn and neqn:

-d*xy* Sets equation delimiters to characters *x* and *y* with the command-line argument. The more common way to do this is with delim*xy* between .EQ and .EN. The left and right delimiters may be identical. Delimiters are turned off by delim off appearing in the text. All text that is neither between delimiters nor between .EQ and .EN is passed through untouched.

-f*n* Changes font to *n* globally in the document. The font can also be changed globally in the body of the document by using the gfont directive.

-p*n* Reduces subscripts and superscripts by *n* point sizes from the previous size. In the absence of the -p option, subscripts and superscripts are reduced by 3 point sizes from the previous size.

-s*n* Sets equations in point size *n* globally in the document. The point size can also be changed globally in the body of the document by using the gsize directive.

-T*dev* Prepares output for device *dev*. If no -T option is present, eqn looks at the environment variable TYPESETTER to see what the intended output device is. If no such variable is found in the environment, a system-dependent default device is assumed. Not available using neqn.

filename Specifies the file to be typeset.

eqn Language

Tokens within `eqn` are separated by braces, double quotation marks, tildes, circumflexes, space, tab, or newline characters. Braces `{ }` are used for grouping; generally speaking, anywhere a single character like *x* could appear, a complicated construction enclosed in braces may be used instead. Tilde (`~`) represents a full space in the output, circumflex (`^`) half as much.

Subscripts and superscripts are produced with the keywords `sub` and `sup`. Thus, `x sub i` makes x_i, `a sub i sup 2` produces a_i^2 and `e sup {x sup 2 + y sup 2}` gives $e^{x^2+y^2}$.

Fractions are made with the keyword `over`: `a over b` yields $\frac{a}{b}$.

Square roots are made with the keyword `sqrt`: `1 over down 10 sqrt {ax sup 2 +bx+c}` results in

$$\frac{1}{\sqrt{ax^2+bx+c}}.$$

Although `eqn` tries to get most things at the right place on the paper, occasionally you will need to tune the output to make it just right. In the previous example, a local motion, *down 10*, was used to get more space between the square root and the line above it.

The keywords `from` and `to` introduce lower and upper limits on arbitrary things: `lim from {n-> inf } sum from 0 to n x sub i` produces $\lim_{n\to\infty}\sum_{0}^{n} x_i$.

Left and right brackets, braces, and so on, of the right height made with the keywords `left` and `right`: `left [ x sup 2 + y sup 2 over alpha right ] ~=~1` produces

$$\left[x^2+\frac{y^2}{\alpha} \right] = 1.$$

The `right` clause is optional. Legal characters after `left` and `right` are braces, brackets, bars, `c` and `f` for ceiling and floor, and `""` for nothing at all (useful for a right-side-only bracket).

Vertical piles of things are made with the keywords `pile`, `lpile`, `cpile`, and `rpile`: `pile {a above b above c}` produces $\begin{matrix} a \\ b \\ c \end{matrix}$.

There can be an arbitrary number of elements in a pile. `lpile` left justifies, `pile` and `cpile` center, with different vertical spacing, and `rpile` right justifies.

Matrices are made with the keyword `matrix`: `matrix { lcol { x sub i above y sub 2 } ccol { 1 above 2 } }` produces

$$\begin{matrix} x_i & 1 \\ y_2 & 2 \end{matrix}.$$

In addition, there is the `rcol` keyword for a right-justified column.

Diacritical marks are made with the keywords `dot`, `dotdot`, `hat`, `tilde`, `bar`, `vec`, `dyad`, and `under`: `x dot = f(t) bar` produces $\dot{x} = \overline{f(t)}$, `y dotdot bar ~=~ n under` produces $\ddot{\bar{y}} = \underline{n}$, and `x vec ~=~ y dyad` produces $\vec{x} = \vec{\vec{y}}$.

Sizes and font can be changed with `size` *n* or `size` ±*n*, `roman`, `italic`, `bold`, and `font` *n*. Size and fonts can be changed globally in a document by `gsize` *n* and `gfont` *n*, or by the command-line arguments `-s`*n* and `-f`*n*.

Successive display arguments can be lined up. Place `mark` before the desired lineup point in the first equation; place `lineup` at the place that is to line up vertically in subsequent equations.

Shorthands may be defined or existing keywords redefined with `define`:

`define` *thing* `%` *replacement* `%`

This example shows a new token called *thing* that will be replaced by *replacement* whenever it appears thereafter. The `%` may be any character that does not occur in *replacement*.

Keywords like `sum` (∑), `int` (∫), `inf` (∞), and shorthands like `>=` (≥), `->` (→), and `!=` (≠) are recognized. Greek letters are spelled out in the desired case, as in `alpha` or `GAMMA`. Mathematical words like sin, cos, and log are made Roman automatically. `troff`(1) four-character escapes, like \(bu (•), can be used anywhere. Strings enclosed in double quotation marks `"`...`"` are passed through untouched; this permits keywords to be entered as text, and can be used to communicate with `troff`(1) when all else fails.

NOTES

To make digits, parentheses, and so on, appear in bold, enclose them in quotation marks, as in `bold "12.3"`.

EXAMPLES

```
eqn filename ... | troff

neqn filename ... | nroff
```

SEE ALSO

`nroff`(1), `tbl`(1), `troff`(1)

`eqnchar`(7D), `ms`(7D) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

NAME

etags – Generates tag file for Emacs

SYNOPSIS

etags [-CDSTVHadt] [-i *file*] [-o *outfile*] [-defines] [-no-defines] [-c++] [-typedefs] [-typedefs-and-c++] [-ignore-indentation] [-help] [-version] [-include=*file*] [-output=*outfile*] [-append] *file* ...

IMPLEMENTATION

All Cray Research Systems

DESCRIPTION

The etags program is used to create a tag table file, in a format understood by emacs(1). This program understands the syntax of C, Fortran, LaTeX, Scheme and Emacs Lisp/Common Lisp. It reads the files specified on the command line, and writes a tag table (default is TAGS) in the current working directory. The programs recognize the language used in an input file based on its file name and contents; there are no switches for specifying the language.

etags accepts the following options and unambiguous abbreviations for long option names.

-d, -defines
: Creates tag entries for C preprocessor definitions. This is the default.

-D, -no-defines
: Does not create tag entries for C preprocessor definitions.

-C, -c++
: Treats files with .c and .h extensions as C++ code, not C code. Files with .C, .H, .cxx, .hxx, or .cc extensions are always assumed to be C++ code.

-t, -typedefs
: Records typedefs in C code as tags.

-T, -typedefs-and-c++
: Generates tag entries for typedefs, struct, enum, and union tags, and C++ member functions.

-S, -ignore-indentation
: Does not assume that a closing brace in the first column is the final brace of a function or structure definition.

-H, -help
: Prints usage information.

-V, -version
: Prints the current version of the program.

-i *file*, -include=*file*
: Includes a note in tag file indicating that, when searching for a tag, one should also consult the tags file *file* after checking the current file.

-o *outfile*, -output=*outfile*
: Explicit name of file for tag table (ignored with -v or -x).

-a, -append
: Appends to existing tag file.

file Name of file to be used to generate tag file.

SEE ALSO

`cxref`(1), `emacs`(1), `vgrind`(1)

NAME

ex – Text editor

SYNOPSIS

ex [-C] [-r] [-R] [-l] [-L] [-V] [-s] [-c *command*] [-t *tag*] [-w *n*] [-x] [*files*]
ex [-C] [-r] [-R] [-l] [-L] [-V] [-v] [-c *command*] [-t *tag*] [-w *n*] [-x] [*files*]

Obsolescent version; may not be supported in future releases:

ex [-C] [-r] [-R] [-l] [-L] [-V] [-] [+ *command*] [-t *tag*] [-w *n*] [-x] [*files*]
ex [-C] [-r] [-R] [-l] [-L] [-V] [-v] [+ *command*] [-t *tag*] [-w *n*] [-x] [*files*]

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX
AT&T extensions (-C, -l, -L, -V and -x options)

DESCRIPTION

The ex editor is the root of a family of editors: ex and vi(1). ex is a superset of ed(1), with the most notable extension being a display editing facility. Display-based editing is the focus of vi.

If you have a CRT terminal, you may want to use a display-based editor; in this case, see vi(1), which is a utility that focuses on the display editing portion of ex.

For ed Users

If you have used ed(1), you will find that ex has several new features useful on CRT terminals. Generally, the ex editor uses far more of the capabilities of terminals than ed(1) does and uses the terminal capability database (see the terminfo(5) man page) and the type of the terminal you are using from the TERM environment variable in the environment to determine how to drive your terminal efficiently. The editor makes use of features such as insert and delete character and line in its visual command (which can be abbreviated vi) and which is the central mode of editing when using vi(1).

The ex editor contains several new features to easily view the text of the file. The z command gives easy access to windows of text. Press <CONTROL-d> to scroll a half-window of text. This is more useful for stepping quickly through a file than just pressing <RETURN>. Of course, the screen-oriented visual mode gives constant access to editing context.

The ex editor gives you more help when you make mistakes. The undo (u) command lets you reverse any single change that has undesired results. ex gives you a lot of feedback, normally printing changed lines, and indicates when more than a few lines are affected by a command so that it is easy to detect when a command has affected more lines than it should.

The ex editor also normally prevents you from overwriting existing files. If the system (or editor) crashes, or if you accidentally hang up the telephone, you can use the recover command to retrieve your work. This command returns you back to within a few lines of where you stopped.

The ex editor has several features for dealing with more than one file at a time. You can specify a list of files on the command line and use the next (n) command to deal with each file in turn. You can also give the next command a list of file names or a pattern, as used by the shell, to specify a new set of files to be edited. In general, file names in the editor may be formed with full shell metasyntax. Metacharacter % is also available in forming file names and is replaced by the name of the current file.

For moving text between files and within a file, the editor has a group of buffers, named a through z. You can place text in these named buffers and carry it over when you edit another file.

The & command in ex repeats the last substitute command. In addition, the confirmed substitute command exists. You specify a range of substitutions to be made, and the editor interactively asks whether each substitution is desired.

You can ignore the case of letters in searches and substitutions. ex also allows you to construct regular expressions that match words to be constructed. This is convenient, for example, in searching for the word edit if your document also contains the word editor.

The ex editor has a set of options that you can set to tailor it to your liking. One very useful option is the autoindent option, which allows the editor to supply leading white space automatically to align text. You can then use the <CONTROL-d> key as a backtab and space and tab forward to align new code easily.

Miscellaneous new useful features include an intelligent join (j) command, which supplies white space between joined lines automatically; commands < and >, which shift groups of lines; and the ability to filter parts of the buffer through commands such as sort.

The ex editor accepts the following options and operands:

-c *command*
+ *command* (Obsolescent)
Begins editing by executing the specified editor search or positioning *command*.

-C Same as -x but assumes *file* began in encrypted form.

-l Sets lisp mode.

-L Prints a list of all recoverable files. Same as -r with no *file* specified.

-V Sets verbose mode.

-r Recovers *file* after an editor or system crash. If no *file* is given, a list of all recoverable files available to the user are written.

-R Sets readonly mode; prevents accidental overwriting of a file.

-s
- (Obsolescent)
Suppresses all interactive-user feedback. This is useful in processing editor scripts.

`-t` *tag*	Edits the file that contains the *tag* and positions the editor at its definition.
`-v`	Invokes `vi`.
`-w` *n*	Sets the value of the window editor option to *n*.
`-x`	Sets encryption option. When this option is used, the file is encrypted as it is written and will require an encryption key to be read (see `crypt`(1)). Also, see the WARNINGS section.
files	Specifies one or more path names of files to be edited.

ex States

Command	Normal and initial state. A colon (`:`) prompts for input. Your kill character cancels partial command.
Insert	Entered by `a`, `i`, or `c`. You may enter arbitrary text. Insert is terminated normally by a line having only `.` on it, or abnormally with an interrupt.
Visual	Entered by `vi`, terminates with `Q` or ^\.

ex Command Names and Abbreviations

abbrev	ab	next	n	undo	u
append	a	number	nu	unmap	unm
args	ar	preserve	pre	version	
change	c	print	p	visual	vi
copy	co	put	pu	write	w
delete	d	quit	q	xit	x
edit	e	read	re	yank	ya
file	f	recover	rec	window	z
global	g	rewind	rew	escape	!
insert	i	set	se	lshift	<
join	j	shell	sh	print next	<return>
list	l	source	so	resubst	&
map	map	stop	st	rshift	>
mark	ma	substitute	s	scroll	^D
move	m	unabbrev	una		

ex Command Addresses

n	Line *n*	/*pat*	Next with *pat*
.	Current	?*pat*	Previous with *pat*
$	Last	*x*-*n*	*n* before *x*
+	Next	*x*,*y*	*x* through *y*
-	Previous	´*x*	Marked with *x*
+*n*	*n* forward	´´	Previous context
%	1,$		

Initializing Options

`EXINIT`	Places `set`'s here in environment variable
`$HOME/.exrc`	Editor initialization file
`./.exrc`	Editor initialization file
`set` *x*	Enable option
`set no`*x*	Disable option
`set` *x*=*val*	Gives value *val*
`set`	Shows changed options
`set all`	Shows all options
`set` *x*?	Shows value of option *x*

Most Useful Options

`autoindent`	`ai`	Supplies indent
`autoprint`	`ap`	Prints current line after buffer changed
`autowrite`	`aw`	Writes before changing files
`beautify`	`bf`	Discards all control characters other than tab, newline, and form-feed
`directory`	`dir`	Specifies the directory
`edcompatible`	`ed`	Causes suffixes to be remembered
`flash`		Flashes screen on errors
`hardtabs`		Value of any hardware tab settings
`ignorecase`	`ic`	Ignore case
`lisp`		Modifies commands for lisp code
`list`		Prints `^I` for tab, `$` at end
`magic`		`. [ *` special in patterns
`mesg`		Allows non-`vi` messages to appear on screen during `vi`
`number`	`nu`	Numbers lines
`paragraphs`	`para`	Macro names that start ...
`prompt`		When set, prompted with a colon (:)
`readonly`		Disallows writing buffer contents to current file name
`redraw`		Simulates smart terminal
`remap`		Macro translation
`report`		Indicates number of lines that must be changed
`scroll`		Command mode lines
`sections`	`sect`	Macro names ...
`shell`		Shell executed when doing :! or !
`shiftwidth`	`sw`	For `< >`, and input `^D`
`showmatch`	`sm`	To `)` and `}` as typed
`showmode`	`smd`	Shows insert mode in `vi`
`slowopen`	`slow`	Stops updates during insert
`tabstop`	`ts`	Software tab stops
`taglength`		Tag names (labels) only significant to this many characters
`tags`		List of files to be searched by the `:tag` command
`term`		Type of terminal you are using

terse		Shortens error messages
timeout		Must enter a macro name in less than 1 second
warn		Gives warning when :! is issued
window		Visual mode lines
wrapscan	ws	Around end of buffer?
wrapmargin	wm	Splits line automatically
writeany	wa	Allows a write to any file

Scanning Pattern Formation

^	Beginning-of-line
$	End-of-line
.	Any character
\<	Beginning-of-word
\>	End-of-word
[*str*]	Any character in *str*
[^*str*]	... not in *str*
[*x*-*y*]	... between *x* and *y*
*	Any number of preceding

WARNINGS

Inclusion of the Data Encryption Standard (DES) encryption code requires a special license for sites outside the United States and Canada. If these encryption functions are not available on your system, check with your system administrator or site analyst.

Files that were encrypted on UNICOS systems prior to release 5.0 must be re-encrypted using crypt(1) before ex -x can read them.

EXIT STATUS

The ex utility exits with one of the following values:

0 Successful completion.

>0 An error occurred.

BUGS

The undo command causes all marks to be lost on lines changed and then restored if the marked lines were changed.

The undo command never clears the buffer modified condition.

The z command prints a number of logical lines rather than physical lines. More than a screenful of output may result if long lines are present.

File input/output errors do not print a name if the command-line - option is used.

There is no easy way to do a single scan that ignores case.

The editor does not warn you when text is placed in named buffers and not used before exiting the editor.

Null characters are discarded in input files and cannot appear in resultant files.

FILES

`/usr/lib/exstrings`	Error messages
`/usr/lib/exrecover`	Recovers command
`/usr/lib/expreserve`	Preserves command
`/usr/lib/terminfo`	Describes capabilities of terminals
`HOME/.exrc`	Editor start-up file
`./.exrc`	Editor start-up file
`TMPDIR/Ex`*nnnnn*	Editor temporary
`TMPDIR/Rx`*nnnnn*	Named buffer temporary
`/usr/preserve/`*login*	Preservation directory (where *login* is the user's login)

SEE ALSO

`awk`(1), `crypt`(1), `ed`(1), `edit`(1), `emacs`(1), `grep`(1), `sed`(1), `vi`(1)

`curses`(3) (available only online)

`term`(5), `terminfo`(5) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

Learning the vi Editor, Linda Lamb, O'Reilly & Associates, Inc., 1990

NAME

exdf – Transfers files to or from the IOS partition on the system console (expander disk for the CRAY EL series)

SYNOPSIS

```
exdf -i directory/file [-r] [-s]
exdf -o directory/file [-r] [-s]
```

IMPLEMENTATION

CRAY J90 series
CRAY EL series

DESCRIPTION

The exdf command can read or write a file on the IOS disk. The program determines from the command line whether it is reading or writing a file.

The exdf command accepts the following options:

-i Indicates input from a file.

-o Indicates output to a file.

-r Indicates that a file being output can replace an existing file. The default is to abort a transfer file if the file already exists. The option is ignored for files being read.

-s Turns off warning messages (the silent switch).

directory/file
Specifies the directory/file name entry as known to the IOS. For all CRAY EL systems, IOS directory file name combinations are automatically forced to uppercase letters. The CRAY J90 IOS is case-sensitive. Standard input or output is used for the UNICOS file. UNICOS directory and file names can be specified through the normal redirect methods.

The exdf command can easily be used as a link in a pipe to move files onto or off of the IOS disk.

WARNINGS

For the CRAY EL IOS, directory and file names must be specified in uppercase. For both the CRAY EL IOS and CRAY J90 IOS, file names must be separated by commas.

BUGS

The IOS and Cray Research systems force full words to be transferred, resulting in null characters added to the end of files. This is a problem with text files that you want to compile. The C preprocessor blows up when it runs into the null characters. exdf cannot filter the data because the last bytes in a file are not always accompanied by the end-of-file status.

Character files on the IOS do not have carriage return and line-feed characters as end-of-lines, but you cannot edit text files created on the IOS.

If `exdf` cannot write a file to the IOS disk because the disk is write-protected, no error messages are displayed.

NAME

expand, unexpand – Expands tabs to spaces and vice versa

SYNOPSIS

expand [-t *tablist*] *files*

unexpand [-a] [*files*]
unexpand [-t *tablist*] [*files*]

Obsolescent version; may not be supported in future releases:

expand [-*tabstop*] [-*tab1,tab2,...,tabn*] [*files*]

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The expand utility writes files or the standard input to the standard output with <tab> characters replaced with one or more <space> characters that are needed to pad to the next tab stop. expand preserves backspace characters into the output and decrements the column count for tab calculations. expand is useful for preprocessing character files (for example, before sorting and looking at specific columns) that contain <tab>s.

The expand utility accepts the following options:

-t *tablist*
-*tabstop* (Obsolescent)
-*tab1,tab2,...,tabn* (Obsolescent)

Specifies the tab stops. If the *tablist* argument is a single decimal integer, tabs are set *tablist* column positions apart, rather than of the default 8. If you specify multiple numbers, the tabs are set at those specific column positions. Multiple numbers are specified as one argument, each number separated with a comma or <blank>s.

In the obsolescent version, the single number is specified as *tabstop* with a leading minus; multiple tab stops are specified after a leading minus as *tab1*, *tab2*, and so on.

The unexpand utility copies files or standard input to standard output, converting <blank>s at the beginning of each line into the maximum number of <tab>s followed by the minimum number of <space>s needed to fill the same column positions originally filled by the translated <blank>s. By default, tabstops are set at every eighth column position.

The unexpand utility accepts the following options:

`-a` In addition to translating <blank>s at the beginning of each line, translates all sequences of two or more <blank>s immediately preceding a tab stop to the maximum number of <tab>s, followed by the minimum number of <space>s that are needed to fill the same column positions originally filled by the translated <blank>s.

`-t` *tablist* Specifies the tab stops. If the *tablist* argument is a single decimal integer, tabs are set *tablist* column positions apart instead of the default 8. If you specifiy multiple numbers, the tabs are set at those specific column positions. Multiple numbers are specified as one argument, each number separated with a comma or <blank>s.

No <space>-to-<tab> conversions occur for characters at positions beyond the last of those specified in a multiple tab-stop list.

When you specify `-t`, any `-a` option is ignored.

files Names of input files you specify.

EXIT STATUS

The expand and unexpand utilities exit with one of the following values:

0 Successful completion.

>0 An error occurred.

NAME

`explain` – Displays the explanation for an error message

SYNOPSIS

`explain` *msgid*

IMPLEMENTATION

UNICOS systems

IRIX systems

DESCRIPTION

The `explain` utility retrieves and outputs a message explanation from an online explanation catalog. If the output device is a terminal, the output of `explain` is piped through the pager specified in the `PAGER` variable. If `PAGER` is not specified, the default pager `more -s` is used. If the output device is not a terminal, the output of `explain` is sent to the standard output device (`stdout`).

The `explain` utility requires the following argument:

msgid Specifies the message ID string associated with a message that appears when an error message is output. This string consists of the product group code and the message number. The product group code (*group*) is a string that identifies the product issuing the message. The message number (*msg#*) specifies which message within the product you have received. Enter the message ID as an argument to `explain` in the form *groupmsg#* or in the form *group-msg#*. If the group code ends in one or more digits (for example, `cf90`), you must use the form that includes the dash (`-`).

Recognized Group Codes

The following tables show the products and group codes that have message and explanation catalogs on UNICOS systems and on IRIX systems. The first column lists the group code (needed to look up the explanations for messages); the second column gives the complete name of the software product or products associated with the group code; the third column lists the publication number in which explanations for messages in that group code can be found.

UNICOS group codes

Group Code	Software Product	Publication
`apprentice`	MPP Apprentice tool	Online only
`as`	CAL assembler	SR–3108
`asm`	Cray Assembler for MPP (CAM)	Online only
`builddefs`	`builddefs`(1) command of the UNICOS glossary	Online only
`cc`	Cray Standard C compiler	Online only
`cf90`	CF90 Fortran language	Online only
`cld`	Loader for CRAY T3E systems	Online only

UNICOS group codes (continued)

Group Code	Software Product	Publication
cmd	UNICOS utilities that are compliant with XPG4	Online only
define	define(1) command of the UNICOS glossary	Online only
deplib	deplib(1) command	Online only
df	df(1) utility	Online only
dm	Data Migration Facility	SG–2135
du	du(1) utility	Online only
ex	ex(1) and vi(1) utilities	Online only
fsquota	File system quota feature	Online only
hpm	hpm(1) performance tool	Online only
hpmall	hpmall(8) performance tool	Online only
hpmflop	hpmflop(8) performance tool	Online only
ldr	segldr(1) and ld(1) loader commands	SR–0066
lib	Fortran and I/O libraries	Online only
libm	Mathematical libraries	SG–2138
libp	Pascal library	Online only
lprof	libprof library	Online only
ltrace	Jumptrace library	Online only
mppldr	Loader for CRAY T3D systems	Online only
mtdump	mtdump(1) performance tool	Online only
nqs	Network Queuing System	Online only
perf	libperf library	Online only
proc	procstat(1) performance tool	Online only
ps	ps(1) utility	Online only
sh	Standard shell	Online only
sys	UNICOS system error list (syserrlist[])	SR–2012
syslog	System message logger (newsys(8) and syslogd(8))	Online only
talk	talk(1B) utility	Online only
urm	Unified Resource Manager	Online only
usm	UNICOS Source Manager (USM)	SG–2097

IRIX group codes

Group Code	Software Product	Publication
cf90	f90 version 7.2	Online only
lib	Fortran 90 library version 2.0, and libffio	Online only
msgsys	explain(1) and caterr(1) utilities	Online only

Location of Explanation Catalogs

To find the explanation for the message, `explain` searches the path specified in the `NLSPATH` environment variable for the *group*`.exp` file. The value of the `NLSPATH` variable depends on the `LANG` environment variable or the `LC_MESSAGES` category. If `explain` cannot access an existing explanation catalog, check the value of the `NLSPATH` environment variable and either the `LANG` environment variable or the `LC_MESSAGES` category to ensure that the path name of the catalog is in the message system search path.

On UNICOS systems, you can use the `whichcat`(1) utility to determine which catalog is being accessed by `explain`. For more information, see `whichcat`(1).

On UNICOS systems, see the `catopen`(3C) man page for a description of the `NLSPATH`, `LANG`, and `LC_MESSAGES` components.

On IRIX systems, see `environ`(5) for a description of the `NLSPATH`, `LANG`, and `LC_MESSAGES` components.

Message Format Variables

If *msgid* does not appear with the error message or if you want to change the format of the messages you receive, you can modify your `MSG_FORMAT` and `CMDMSG_FORMAT` environment variables. Both variables can be set to define the fields and the order of the fields to be included in message output.

The `MSG_FORMAT` variable controls the format in which you receive error messages from programs that use the message formatting function `catmsgfmt`(3C). On UNICOS systems, these are most messages in group codes other than `cmd`. On IRIX systems, these are most messages in group codes other than `msgsys`.

On UNICOS systems, the `CMDMSG_FORMAT` variable controls the format in which you receive error messages issued by system utilities that are included in the XPG4 standard. These are messages in the `cmd` group code. A separate environment variable is provided to control message output from these utilities, because it is often desirable to capture and process this output through a script or program; limiting the output to one line makes this easier. The default format (shown below) for non-utility messages outputs at least two lines. The `CMDMSG_FORMAT` variable exists to provide a one-line default format for utility messages.

On IRIX systems, the `explain` and `caterr` utilities use either `CMDMSG_FORMAT` or `MSG_FORMAT` variables; however, `CMDMSG_FORMAT` is used before `MSG_FORMAT`.

Message Format Syntax

Valid fields for `MSG_FORMAT` and `CMDMSG_FORMAT` are as follows:

`%G` Group code

`%N` Message number

`%C` Command name

`%S` Severity level

`%P` Position of the error

`%M` Message text

`%D` Debugging information

`%T` Time stamp

If one of the % fields is not present in the contents of `MSG_FORMAT` or `CMDMSG_FORMAT`, the corresponding message field is not printed.

The default message format is produced by the following assumed `MSG_FORMAT` contents:

```
%G-%N  %C: %S %P\n  %M\n
```

For messages issued by UNICOS utilities, and by the IRIX utilities `explain`(1) and `caterr`(1), the default message format is produced by the following assumed `CMDMSG_FORMAT` contents:

```
%G-%N: %C %M\n
```

Messages issued from the `cmd` group code determine their format according to the following precedence:

1. Content of `CMDMSG_FORMAT` variable, if it exists
2. Content of `MSG_FORMAT` variable, if it exists
3. Default message format for utilities

The format of the time stamp (`%T`) is equivalent to that produced by the `cftime`(3C) function and can be overridden by the `CFTIME` environment variable. For details about time stamp formats, see the `strftime`(3C) man page, which documents the `cftime` function.

The Standard C `printf` escape sequences also can be embedded in the contents of `MSG_FORMAT`. The following table lists special characters that can be embedded.

Description	Symbol	Sequence
New-line character	NL (LF)	`\n`
Horizontal tab	HT	`\t`
Vertical tab	VT	`\v`
Backspace	BS	`\b`
Carriage return	CR	`\r`
Form feed	FF	`\f`
Audible alert	BEL	`\a`
Backslash	\	`\\`
Question mark	?	`\?`
Single quotation mark	’	`\’`
Double quotation mark	"	`\"`
Octal number	*ooo*	`\`*ooo*
Hexadecimal number	*hh*	`\x`*hh*

The escape *ooo* consists of the backslash followed by 1, 2, or 3 octal digits, which are taken to specify the value of the desired character. A common example of this construction is `\0`, which specifies the null character. The escape `\x`*hh* consists of the backslash, followed by `x`, followed by hexadecimal digits, which are taken to specify the value of the desired character. There is no limit on the number of digits, but the behavior is undefined if the resulting character value exceeds that of the largest character.

Any characters other than those listed in this table are passed through without the backslash, (for example, `\q` produces `q`).

NOTES

On UNICOS systems, if this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
`system`, `secadm`	In a privileged administrator shell environment, allowed to write shell-redirected output to any file.
`sysadm`	Shell-redirected output is subject to security label restrictions.

If the `PRIV_SU` configuration option is enabled, the super user can write shell-redirected output to any file.

SEE ALSO

UNICOS systems

`caterr`(1), `catxt`(1), `gencat`(1), `whichcat`(1)

`catgetmsg`(3C), `catgets`(3C), `catmsgfmt`(3C), `catopen`(3C), `strftime`(3C) in the *UNICOS System Libraries Reference Manual*, Cray Research publication SR–2080

`nl_types`(5), `msg`(7D) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

IRIX systems

`caterr`(1), `gencat`(1)

`catgetmsg`(3C), `catmsgfmt`(3C)

`msg`(7D)

Cray Research publications

Cray Message System Programmer's Guide, Cray Research publication SG–2121

Segment Loader (SEGLDR) and ld Reference Manual, Cray Research publication SR–0066

UNICOS System Calls Reference Manual, Cray Research publication SR–2012

UNICOS Source Manager (USM) User's Guide, Cray Research publication SG–2097

Cray Data Migration Facility (DMF) Administrator's Guide, Cray Research publication SG–2135

Intrinsic Procedures Reference Manual, Cray Research publication SR–2138

Cray Assembly Language (CAL) for Cray PVP Systems Reference Manual, Cray Research publication SR–3108

NAME

`expr` – Evaluates arguments as an expression

SYNOPSIS

`expr` *arguments*

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The `expr` utility evaluates an expression and writes the result to standard output. All *arguments* are taken as expressions. After evaluation, the result is written on standard output. You must express the terms of the expression with blanks and escape characters special to the shell. Note that 0 is returned to indicate a value of 0, rather than the null string. Strings containing blanks or other special characters should be enclosed in quotation marks. Integer-valued arguments can be preceded by a unary minus sign. Internally, integers are treated as 64-bit, twos-complement numbers. The length of the expression is limited to 512 characters.

The following is a list of operators and keywords. Characters that must be escaped are preceded by `\`. The list is in order of increasing precedence, with equal-precedence operators grouped between `{}` symbols.

Grouping may be accomplished for the purpose of precedence control by use of the `\(` and `\)` operators. `y = (a+b)/c` becomes `expr \($a+$b\)/$c`.

`expr | expr`	Returns the first `expr` when it is neither null nor 0; otherwise, it returns the second `expr`.
`expr \& expr`	Returns the first `expr` when `expr` is neither null nor 0; otherwise, it returns 0.
`expr` { `=`, `\>`, `\>=`, `\<`, `\<=`, `!=` } `expr`	Returns the result of an integer comparison when both arguments are integers; otherwise, it returns the result of a lexical comparison.
`expr` { `+`, `-` } `expr`	Addition or subtraction of integer-valued arguments.
`expr` { `\*`, `/`, `%` } `expr`	Multiplication, division, or remainder of the integer-valued arguments.
`expr` : `expr`	Matching operator `:` compares the first argument with the second argument, which must be a regular expression. Regular expression syntax is the same as that of `ed`(1), except that all patterns are ‘‘anchored’’ to the beginning of the line (that is, begin with `^`). Therefore, `^` is not a special character in that context. Usually, the matching operator returns the number of characters matched (0 on failure). Alternatively, you can use the `\(...\)` pattern symbols to return part of the first argument.

NOTES

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
`system`, `secadm`	In a privileged administrator shell environment, allowed to write shell-redirected output to any file.
`sysadm`	Shell-redirected output is subject to security label restrictions.

If the `PRIV_SU` configuration option is enabled, the super user can write shell-redirected output to any file.

EXIT STATUS

As a side effect of expression evaluation, `expr` returns the following exit values:

0 Expression is neither null nor 0, or expression is false.

1 Expression is null or 0, or expression is true.

2 Expressions are not valid.

MESSAGES

`syntax error`	Operator or operand errors.
`nonnumeric argument`	Arithmetic is tried on a noninteger string.

BUGS

After argument processing by the shell, `expr` cannot tell the difference between an operator and an operand except by the value. If `$a` is `=`, the command:

```
expr $a = '='
```

looks like the following, because the arguments are passed to `expr` (and they will all be taken as the `=` operator):

```
expr = = =
```

The following command works:

```
expr X$a = X=
```

EXAMPLES

Example 1: The following example adds 1 to standard shell variable `a`.

```
a=‘expr $a + 1‘
```

Example 2: The following example adds 1 to C shell variable a.

```
set a=‘expr $a + 1‘
```

Example 3: The following example returns the last segment of a path name (that is, *file*). Watch out for / alone as an argument; expr takes it as the division operator (see the BUGS section).

```
# ’For $a equal to either "/usr/abc/file" or just "file"’
expr $a : ’.*/\(.*\)’ \| $a
```

The addition of the // characters eliminates any ambiguity about the division operator and simplifies the whole expression.

```
#  A better representation of the previous example.
expr //$a : ’.*/\(.*\)’
```

Example 4: The following example returns the number of characters in $VAR.

```
expr $VAR : ’.*’
```

SEE ALSO

csh(1), ed(1), sh(1)

regexp.h(3C) in the *UNICOS System Libraries Reference Manual*, Cray Research publication SR–2080

NAME

`factor` – Factors a number

SYNOPSIS

`factor` [*number*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

When `factor` is invoked without an argument, it waits for a number to be typed in. If you type in a positive number less than or equal to 1.0×10^{14} it will factor the number and print its prime factors; each one is printed the proper number of times. Then it waits for another number. If it encounters a 0 or any nonnumeric character, it exits.

If `factor` is invoked with an argument, it factors the number as above and then exits.

Maximum time to factor is proportional to $\sqrt{n}$ and occurs when n is prime or the square of a prime.

The `factor` utility requires the following option:

number The input number you specify.

MESSAGES

`Ouch!` Input out of range or garbage input

BUGS

Garbage input is not always diagnosed (for example, `factor hello`).

NAME

fc – Displays process command history list

SYNOPSIS

```
fc [-r] [-e editor] [first [last] ]
fc -l [-n] [-r] [first [last] ]
fc -s [old=new] [first]
```

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The fc utility lists, or edits and reexecutes, the command previously entered to an interactive shell (see sh(1)).

The command history list references commands by number. The first number in the list is selected arbitrarily. The relationship of a number to its command does not change except when the user logs in and no other process is accessing the list, at which time the system may reset the numbering to start the oldest retained command at another number, usually 1. When the number reaches an upper limit, the shell wraps the numbers, starting the next command with the number fc retains the time-ordering sequence of the commands.

When commands are edited (when you omit the -l option), the line(s) that result are entered at the end of the history list and then reexecuted by the shell. The fc command that caused the editing is not entered into the history list. If the editor returns a nonzero exit status, the entry is not placed into the history list and the command reexecution. Any command-line variable assignments or redirection operators used with fc affect both the fc command itself and the command that results. For example, the following command line reinvokes the previous command, suppressing standard error for both fc and the previous command:

```
fc -s -- -1 2>/dev/null
```

The fc utility accepts following options and operands:

-e *editor* Specifies the *editor* to edit the commands. The *editor* string is a utility name, subject to search by using the PATH environment variable. The value in the FCEDIT variable is used as a default when you omit the -e option. If FCEDIT is null or unset, ed(1) is used as the editor.

-l Lists the commands rather than invoking an editor on them. The commands are written in the sequence indicated by the *first* and *last* operands, as affected by -r, with each command preceded by the command number.

-n Suppresses command numbers when listing with -l.

`-r` Reverses the order of the commands listed (with `-l`) or edited (with neither `-l` or `-s`).

`-s` Reexecutes the command without invoking an editor.

first
last Specifies the commands to list or edit. The value of the `HISTSIZE` environment variable determines the number of previous commands. The value of *first* or *last* or both are one of the following:

[+]*number* A positive number that represents a command number; to display command numbers, use the `-l` option.

`-`*number* A negative decimal number that represents the command that was executed *number* of commands previously. (For example, the `-l` is the immediately previous command).

string A string that indicates the most recently entered command that begins with that string. If you do not also specify the *old=new* operand with `-s`, the string form of the *first* operand cannot contain an embedded equal sign.

When the synopsis form with `-s` is used:

- If you omit *first*, the previous command is used.

For the synopsis forms without `-s`:

- If you omit *last*, *last* defaults to the previous command when you specify `-l`; otherwise, it defaults to *first*.
- If you omit *first* and *last*, the previous sixteen commands are listed or the previous one command is edited (based on the `-l` option).
- If *first* and *last* are both present, all of the commands from *first* to *last* are edited (without `-l`) or listed (with `-l`). To edit multiple commands, present to the editor all of the commands at one time, each command starting on a new line. If *first* represents a newer command than *last*, the commands are listed or edited in reverse sequence, equivalent to using `-r`.
- When you use a range of commands, it is not an error to specify *first* or *last* values that are not in the history list; `fc` substitutes the value that represents the oldest or newest command in the list, as appropriate.

old=new Replaces the first occurrence of string *old* in the commands to be reexecuted by the string *new*.

NOTES

The `fc` utility described in the man page is a built-in utility to the standard shell (`sh`(1)). An executable version of this utility is available in `/usr/bin/fc`.

ENVIRONMENT VARIABLES

FCEDIT — Interprets the variable, when expanded by the shell, as the default value for the `-e` *editor* argument. If `FCEDIT` is null or unset, `ed`(1) is used as the editor.

HISTFILE — Interprets this variable as a path name specifying a command history file. If the `HISTFILE` is not set, the shell tries to access or create a file `.sh_history` in the user's home directory.

HISTSIZE — Interprets this variable as a decimal number that represents the limit to the number of previous commands that are accessible. If this variable is unset, the default is 128.

EXIT STATUS

The `fc` utility exits with one of the following values:

0 Successful completion of the listing.

>0 An error occurred.

Otherwise, the exit status is that of the commands executed by `fc`.

SEE ALSO

`sh`(1)

NAME

fck – Provides file information from file system internals

SYNOPSIS

fck [-b] [-i] [-l] [-p] *files*

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The fck utility provides information concerning the named files that are gathered from reading the inode and the address blocks from the block special device in the /dev/dsk directory. Information is printed only on files for which you have read permission.

The fck utility accepts the following options:

-b Displays block-oriented logical addressing information. This includes logical device name, slice, block offsets, and lengths for each data and address extent.

-i Displays information returned by the stat(2) system call. This includes file ownership, permissions, length, access and creation times, and so on. This is the default if no other options are specified.

-l Displays addressing information about file addressing and information blocks, as present in the physical disk inode. This includes inode information that is not reported by the stat(2) system call.

-p Displays physical addressing information. This includes physical device name, cylinder, track, and sectors for each data and address extent. On CRAY PVP systems, users who are not super-users are not shown cylinder and track value information. These values are represented by ***.

files Specifies files for which to gather information.

The program uses the access(2) system call to assure that the user has read permission to the file in question.

SEE ALSO

access(2), stat(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

stor(8) in the *UNICOS Administrator Commands Reference Manual*, Cray Research publication SR–2022

NAME

fg – Runs jobs in the foreground

SYNOPSIS

fg [*job_id* ...]

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The fg utility moves a background job from the current shell execution environment (see sh(1)) into the foreground.

Using fg to place a job into the foreground removes its process ID from the list of those "known in the current shell execution environment."

The fg utility supports the following operand:

job_id Specifies the job to be run as a foreground job. If you omit the *job_id*, the *job_id* for the job that was most recently suspended, placed in background, or run as a background job is used. The Jobs subsection in the sh(1) man page describes the the format of *job_id*.

NOTES

The fg utility described in this man page is a built-in utility to the standard shell (sh(1)). An executable version of this utility is available in /usr/bin/fg.

EXIT STATUS

The fg utility exits with one of the following values:

0 Successful completion.

>0 An error occurred.

SEE ALSO

bg(1), jobs(1), sh(1)

NAME

`file` – Determines file type

SYNOPSIS

```
file [-f ffile] [-h] [-m mfile] [-o] file ...
file [-h] [-m mfile] [-o] -f ffile
file -c [-m mfile]
```

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4
AT&T extensions (`-c`, `-h`, `-f`, and `-m` options)
CRI extensions (`-o` option)

DESCRIPTION

The `file` utility performs a series of tests on each *file* and, optionally, on each file supplied in *ffile*, in an attempt to classify them. If a *file* argument seems to be a text file, `file` examines the first 512 bytes and tries to guess its language. If a *file* argument is a symbolic link, by default the link is followed, and `file` tests the file that the symbolic link references.

The `file` utility uses the `/etc/magic` file to identify files that have some sort of magic number; that is, any file contains a numeric or string constant that indicates its type. Commentary at the beginning of `/etc/magic` explains its format.

The `file` utility accepts the following options:

`-c` Causes the `file` utility to check the *magic* file for format errors, and prints on standard output the *magic* file in a readable format. If the `-c` option is specified, all options except [`-m` *mfile*] are ignored. For reasons of efficiency, this validation is not usually performed.

`-f` *ffile* Indicates that the next argument is a file that contains the names of files to be examined.

`-h` Instructs the `file` utility to not follow symbolic links. If you specify the `-h` option and *file* is a symbolic link, `file` prints the following message:

```
symbolic link to filename
```

The operand *filename* refers to the contents of the symbolic link.

`-m` *mfile* Instructs the `file` utility to use an alternate *magic* file.

`-o` Causes major and minor device numbers, and device-specific parameters of character and block special devices to be printed in octal. The device-specific parameters are also printed in octal.

file Name of file to be examined.

NOTES

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
`system`, `secadm`	In a privileged administrator shell environment, shell-redirected I/O is not subject to file protections.
`sysadm`	Shell-redirected output is subject to security label restrictions.

If the `PRIV_SU` configuration option is enabled, shell-redirected I/O on behalf of the super user is not subject to file protections.

EXIT STATUS

The `file` utility exits with one of the following values:

0 Successful completion.

>0 An error occurred.

BUGS

The `file` utility may mistakenly classify a text file because a bit map in the text file coincides with a bit map in the `/etc/magic` file.

EXAMPLES

The following example determines the type of files `example.c` and `a.out`:

```
$ file example.c a.out
example.c:  c program text
a.out:      executable not stripped
$
```

FILES

`/etc/magic`	File that contains magic numbers for different file types

SEE ALSO

`a.out`(5), `relo`(5) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

NAME

`find` – Finds files

SYNOPSIS

`find` *pathnames expression*

IMPLEMENTATION

Cray PVP systems

STANDARDS

POSIX, XPG4
AT&T extensions (`-follow`, `-fstype`, `-inum`, `-local`, and `-mount` primaries)
CRI extensions (`-acid` and `-dev` primaries)

DESCRIPTION

The `find` utility locates files and directories that are located under the path names you specify and that meet the characteristics you list in the *expression.*

You can use metacharacters in many of the primaries if the special characters are *quoted*, that is, preceded with a backslash or surrounded by single quotation marks. (For more information on quoting, see `sh`(1).) The `find` utility searches for files that cause the Boolean *expression*, composed of any of the following primaries, to be true. When you specify a value for *n* in these primaries, you can prefix that number with either `+` or `-`. The meanings are as follows:

`+`*n* More than *n*

n Exactly *n*

`-`*n* Less than *n*

The primaries fall into two categories: those that select files and those that perform actions on the selected files.

Primaries That Select Files

CAUTION: For POSIX compliance, UNICOS 9.0 included a change in the functionality of the `-atime`, `-ctime`, and `-mtime` primaries. The meaning of the numeric argument to those primaries changed in two ways:

1. From day units (0 equals the 24 hours since the last midnight) to 24-hour blocks starting with the current time (0 equals the 24-hour block starting now).
2. `+` and `-` now refer to the same starting point in time, instead of `-` being time after the end of the 24-hour block and `+` being time before the start of the 24-hour block.

The primaries that select files are as follows:

pathnames Specifies the path names under which `find` will search. A path name can be either full, such as `/abc/xyz`, or relative, such as . to indicate the current directory.

`-acid` *acctid* Locates files with the specified account ID. You can find the account ID of a file by using the `chacid`(1) utility. If the account has the specified account ID, the expression is true.

`-atime` *n* Locates files that have been accessed in *n* days. The `find` utility itself changes the access time of directories in *pathnames*. If the file has been accessed in *n* days, the expression is true.

`-ctime` *n* Locates files that have been modified or whose attributes have been changed. A file's attributes include, for example, its user ID or group ID or the number of links to true. If the file has been changed in the last *n* days, the expression is true.

`-depth` Descends the directory hierarchy. All entries in a directory are acted on before the directory itself. This can be useful when `find` is used with `cpio`(1) to transfer files contained in directories without write permission. Always true.

`-dev` *minor* Locates files that reside on the device with the specified number. If the file resides on the device with minor device number *minor*, it is true.

`-fstype` *type* True if the file system to which the file belongs is of type *type*. Valid *type*s include: `DFS`, `NC1FS`, `NFS`, `PROC`, and `SFS`.

`-group` *gname* Locates files belonging to the specified group. If *gname* is numeric and is not one of the groups listed in the file `/etc/group`, it is taken as a group ID. True if the file belongs to the group *gname*.

`-inum` *ino* Locates files that have only the specified inode number. If the file being processed has an inode number of *ino*, it is true.

`-links` *n* Locates files that have *n* links. If the file being processed has *n* links, it is true.

`-local` Locates files that reside only on the local system. If the file being processed physically resides on the local system, it is true.

`-mount`
`-xdev` Restricts the search to the file system that contains the directory specified. Always true.

`-mtime` *n* Locates files that have been modified in the past *n* days. If the file being processed has been modified in the past *n* days, it is true.

`-name` *file* Locates files that have the specified basename of the file name. You can use metacharacters in the file name if you precede them by \ or surround them with single quotation marks. (Be careful when using `[`, `?`, and `*`.) If *file* matches the file name being processed, it is true.

`-newer` *file* Locates files that have been modified more recently than the specified file. You can use metacharacters in the file name if you precede them by \ or surround them with single quotation marks. If the file being processed has been modified more recently than *file*, it is true.

`-nogroup` True if the file belongs to a group not in the `/etc/group` file.

`-nouser` True if the file belongs to a user not in the `/etc/passwd` file.

`-perm` [`-`]*mode* Locates file that have access permission indicated by the symbolic mode *mode*. The *mode* argument is used to represent file mode bits. It is identical in format to the `chmod` *symbolic mode* operand described in `chmod`(1) and is interpreted as follows. To start, a template is assumed with all file mode bits cleared. An *op* symbol of `+` sets the appropriate mode bits in the template; `-` clears the appropriate bits; `=` sets the appropriate mode bits, without regard to the contents of the file mode creation mask of the process. The *op* symbol of `-` cannot be the first character of *mode*.

If the hyphen is omitted, `-perm` evaluates as true when the file permission bits exactly match the value of the resulting template.

Otherwise, if *mode* is prefixed by a hyphen, the `-perm` evaluates as true if at least all the bits in the resulting template are set in the file permission bits.

`-perm` [`-`]*onum* Locates files that have the access permission indicated by octal number *onum*. (For information about specifying access permissions, see `chmod`(1).) You can prefix *onum* with a hyphen to make more flag bits (07777, see `stat`(2)) significant. True if the file permission flags match octal number *onum*.

`-prune` Always yields true. Do not examine any directories or files in the directory structure below the *pattern* just matched.

`-size` *n*[`c`] Locates files of the specified block length. To specify size in characters, follow the *n* specification with a `c`. If the file being processed is *n* blocks long (512 bytes per block), the expression is true.

`-type` *filetype* Locates files of the specified file type. The file type is specified by one of the following letters:

`b` Block special file
`c` Character special file
`d` Directory
`f` Regular file
`l` Symbolic link
`m` Migrated file (type IFOFL)
`M` Migrated file (has DMF handle)
`p` FIFO or named pipe
`s` Sockets

If *filetype* is `b`, `c`, `d`, `f`, `l`, `m`, `M`, `p`, or `s`, the file type is true.

`-user` *uname* Locates files belonging to the specified user. If *uname* is numeric and is not one of the login names in the file `/etc/udb`, it will be taken as a user ID. True if the file being processed belongs to user *uname*.

Primaries That Act on Files

The primaries that act on files are as follows:

`-exec` *utility_name* [*argument...*] `;`
Executes the specified utility with the given arguments. Follow the command or last argument with a `<space>` and an escaped semicolon (`\;`), as in the following example:

```
find pathnames -exec ls \;
```

`find` replaces a utility argument `{}` with the name of the *pathname* file that is currently being processed. (For more information on parameter substitution with `{}`, see `sh`(1).) If the executed utility *utility_name* returns a value of 0 as the exit status, this expression is true.

If the last argument to `-exec` is `{}`, and you specify '`+`' rather than '`;`', the command will be invoked fewer times with `{}` replaced by groups of pathnames.

`-follow` Follows symbolic links. Always true. When following symbolic links, `find` keeps track of the directories visited to that it can detect infinite loops; for example, such a loop would occur if a symbolic link pointed to an ancestor. This expression should not be used with the `-type l` expression.

`-ok` *utility_name* [*argument...*] `;`
Prompts for execution of the utility with the given arguments. This primary executes *utility_name* as `-exec` does, except that the generated command line is printed with a question mark prompt first, and it is executed only if you respond by typing `y`. If the executed utility returns a value of 0 as the exit status, the expression is true.

`-print` Writes the names of files selected from *pathnames* to `stdout`. Always true.

`(` *expression* `)` If the parenthesized expression is true, the expression is true. Each element in *expression* is a separate argument; you must separate each argument from each other by space characters. A space character must appear on both sides of each parenthesis, exclamation point, or other element (such as a primary or an argument to a primary). When you use a backslash to quote a special character, the space characters go on either side of the pair of characters (for example, " `\[` "). *Expression* is evaluated left to right.

You can combine the primaries by using the following operators (in order of decreasing precedence):

1. The negation of a primary (`!` is the unary NOT operator).
2. Concatenation of primaries (the AND operation is implied by the juxtaposition of two primaries). Any primaries that appear after a primary evaluated as false are false.
3. Alternation of primaries (`-o` is the OR operator).

NOTES

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
`system`, `secadm`	In a privileged administrator shell environment, shell-redirected I/O is not subject to file protections.
`sysadm`	Shell-redirected output is subject to security label restrictions.

If the `PRIV_SU` configuration option is enabled, shell-redirected I/O on behalf of the super user is not subject to file protections.

EXIT STATUS

The `find` utility exits with one of the following values:

0 All *path* operands were traversed successfully.

>0 An error occurred.

EXAMPLES

Example 1: The following command is used to find a file named `data1` under the current directory. The dot character specifies the current directory, the `-name` primary specifies the file `data1` as the file to find, and the `-print` primary causes the output to be displayed on standard output (`stdout`). If `find` does not find the file, no message appears. If `data1` is in the current directory or one of its subdirectories, `find` will display the path to it from the current directory.

```
find . -name data1 -print
```

Example 2: The following command finds files that begin with the letter `b` under the directory `/usr/man`. The output is displayed on `stdout`.

```
find /usr/man -name b'*' -print
```

Example 3: The following example redirects standard error to the null file while finding all files and directories that begin with the letter `b` in the current directory and its subdirectories. This redirection can be a convenient way to separate error messages (for example, those directories to which you may not have access) from the output in which you are interested. This type of redirection of standard error is appropriate for the standard shell.

```
find . -name b'*' -print 2>/dev/null
```

Example 4: The following example removes files named `a.out` and files that end in `.o` that have not been accessed for a week. `-atime +7` indicates a time of 8 or more days, and the final characters instruct `find` to execute the `rm`(1) utility on the selected files.

```
find / \( -name a.out -o -name '*.o'\) -atime +7 -exec rm {} \;
```

FILES

`/etc/group`	Group file that contains group names and group IDs
`/etc/udb`	User validation file that contains user control limits
`/usr/include/sys/fsid.h`	File that contains file system names

SEE ALSO

`chacid`(1), `chmod`(1), `cpio`(1), `rm`(1), `sh`(1), `test`(1)

`statfs`(2), `sysfs`(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

`nftw`(3C) in the *UNICOS System Libraries Reference Manual*, Cray Research publication SR–2080

`cpio`(5), `fs`(5) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

NAME

`finger`, `f` – Provides user information

SYNOPSIS

`/usr/ucb/finger` [`-b`] [`-f`] [`-h`] [`-i`] [`-l`] [`-m`] [`-p`] [`-q`] [`-s`] [`-w`] *names*
`/usr/ucb/finger` [*name*]`@`*host*

`/usr/ucb/f` [`-b`] [`-f`] [`-h`] [`-i`] [`-l`] [`-m`] [`-p`] [`-q`] [`-s`] [`-w`] *names*
`/usr/ucb/f` [*name*]`@`*host*

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

By default, the `finger` and `f` commands (they are functionally equivalent) list the login name, full name, terminal name, write status (as an asterisk (`*`) before the terminal name if write permission is denied), idle time, login time, office location, and phone number (if they are known) for each current user. Idle time is represented in minutes if it is a single integer, hours and minutes if a colon (`:`) is present, or days and hours if a `d` is present in the date field.

These commands have the following options:

`-b` Provides brief list of users.

`-f` Suppresses heading in short and quick output format.

`-h` Suppresses printing of `.project` file.

`-i` Same as quick list but includes idle time.

`-l` Forces long output format.

`-m` Matches arguments only on user name.

`-p` Suppresses printing of `.plan` file.

`-q` Provides quick list with only login name, terminal name, and login time.

`-s` Provides short list of users.

`-w` Suppresses printing of full name in short-list format.

name Name of user you specify.

A longer list format also exists; `finger` and `f` use it whenever a list of names is given. (First and last names of users are accepted.) This is a multiline format, and it includes all information previously described, as well as the user's home directory and login shell, any plan that the person has placed in the `.plan` file in their home directory, and the project on which they are working from the `.project` file, also in the home directory (text must begin on line 1 of the file; if it does not, the command will not read it).

FILES

`/etc/utmp`	File that holds user information
`/etc/udb`	User validation file that contains user control limits
`$HOME/.plan`	Plan file
`$HOME/.project`	Project file

NAME

flodump – Recovers Flowtrace data from a dump or running process

SYNOPSIS

flodump [-e] [-V] [-f *corefile* | -p *procid*]

IMPLEMENTATION

Cray PVP systems

DESCRIPTION

The flodump utility produces Flowtrace output from an abnormally terminated program, compiled using the Cray Research CF90 or Standard C compilers, and with the Flowtrace feature enabled. Programs that make calls to FLOWMARK(3C) are also candidates for this recovery process.

In addition, flodump can capture the Flowtrace data from a running program, if it is owned by the caller to flodump and the running program was compiled with the Flowtrace feature enabled.

The data written by this utility to stdout is raw Flowtrace data, suitable for postprocessing by flowview(1) or by other tools, such as awk(1).

When it is recovering data from a core dump, flodump automatically recovers the data that would have been generated by a nonfatal termination of the program. It reads the file core by default, whose name can be changed by the -f option.

When it is capturing the Flowtrace data from a running process, flodump takes a "snapshot" of the current program state.

Flowtrace is discussed on the man page flowtrace(7); flowview(1) is discussed on its own man page. Both commands are also described in the *Guide to Parallel Vector Applications*, Cray Research publication SG–2182.

The flodump utility writes the captured or recovered data only under the following conditions:

- If your program was compiled with the Flowtrace compiler option turned on, and/or the program executed calls to FLOWMARK
- If the version of Flowtrace loaded with your program was the same as that generated with flodump

The data is less accurate than if the program terminated normally. flodump assumes that the program state is at the last subprogram entry or exit it processed. The distortion introduced by this assumption can occasionally produce abnormal results, such as apparently negative run times.

The flodump utility accepts the following options:

-e Encodes error messages into special raw-data records if errors occur during the data recovery or capture process. Otherwise the default is for flodump to issue error messages and terminate.

`-V` Prints the version and a copyright statement to the `stderr` file.

`-f` *corefile* Specifies a different file name from which to recover the Flowtrace data. By default, if `flodump` is reading a core file, the file name read will be `core`. This option may not be specified at the same time as the `-p` option.

`-p` *procid* Specifies the process number (decimal) of the running program from which to capture the Flowtrace statistics. The program executing in this process must have been compiled with the Flowtrace feature enabled. If this option is not specified, the default is for `flodump` to attempt to recover Flowtrace data from a core dump of an abnormally terminated program. This option cannot be specified at the same time as the `-f` option.

FILES

`core` Program memory dump

SEE ALSO

`flowview`(1)

`cc`(1) in the *Cray Standard C Reference Manual*, Cray Research publication SR–2074

`f90`(1) in *CF90 Commands and Directives Reference Manual*, Cray Research publication SR–3901

`FLOWMARK`(3C) in *UNICOS System Libraries Reference Manual*, Cray Research publication SR–2080

`core`(5) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

`flowtrace`(7), `performance`(7) (available only online)

Guide to Parallel Vector Applications, Cray Research publication SG–2182

NAME

fmgen – Fortran makefile generator

SYNOPSIS

fmgen [-c *compiler*] [-f *flags*] [-m *makefile*] [-o *command_name*] *files*

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The fmgen utility splits out all the Fortran subroutines from *files* using fsplit(1) and produces a makefile using make(1) to compile and load the program.

The fmgen utility accepts the following options:

-c *compiler*	The Fortran compiling system used to convert the .f files to .o files and linking them to form an executable.
-f *flags*	Options to the Fortran compiling system. Be sure to enclose the argument specifying the option(s) in single quotation marks.
-m *makefile*	The name of the makefile produced by the fmgen utility (makefile by default).
-o *command_name*	Resultant executable (a.out by default).
names	Names of input files that you specify.

The makefile created has the following targets:

all	Creates the executable file *command_name* (default rule)
command_name	Compiles and loads the program
command_name.prof	Compiles and loads the program with profiling
clean	Removes all the .o files
clobber	Executes clean and then removes the executable files
void	Removes everything created by the fmgen utility

FILES

makefile	Default make file created
a.out	Executable binary file
a.out.prof	Executable binary file with profiling information
file.f	Input Fortran source code file
file.o	Relocatable object code file

SEE ALSO

f90(1), fsplit(1), make(1), prof(1)

NAME

`fold` – Folds long lines of files for finite-width output device

SYNOPSIS

`fold` [`-b`] [`-s`] [`-w` *width*] [*files*]

Obsolescent version:

`fold` [`-`*width*] [*files*]

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The `fold` utility acts as a filter that splits lines (inserts a `<newline>`) of the specified *files*, or the standard input (if you do not specify *files*), to have maximum width *width*.

The following options are recognized:

`-b` Count *width* in bytes rather than column positions.

`-s` If segment of a line contains a `<blank>` within the first *width* column positions (or bytes), break the line after the last such `<blank>` that meets the width constraints. If no `<blank>` meets the requirements, the `-s` option has no effect for that output segment of the input line.

`-`*width* (obsolescent)
`-w` *width* Specifies the maximum line length (in column positions) (or bytes if `-b` is specified). The default *width* is 80. The output goes to `stdout`. If `<tab>`s are present, or if they must be expanded (using `expand`(1) before using `fold`), the *width* should be a multiple of 8.

files Specifies the path name of a file to be used by this utility.

NOTES

Because multibyte characters are not supported in the UNICOS 8.0 release, the `-b` option assumes 1 byte per character.

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
`system`, `secadm`	Allowed to manage any input file. In a privileged administrator shell environment, shell-redirected I/O is not subject to file protections.

sysadm	Allowed to manage any input file subject to security label restrictions. Shell-redirected I/O is subject to security label restrictions.

If the PRIV_SU configuration option is enabled, the super user is allowed to manage any input file. Shell-redirected I/O on behalf of the super user is not subject to file protections.

EXIT STATUS

The fold utility exits with one of the following values:

0 All input files were processed successfully.

>0 An error occurred.

BUGS

If underlining is present, it may not work correctly with the fold utility.

EXAMPLES

The following command expands all <tab> characters in the file xyz, then creates a xyz.folded file in which the longest line in the file is no longer than 80 characters.

```
expand xyz | fold > xyz.folded
```

SEE ALSO

expand(1) to expand tabs to spaces and vice versa

NAME

`from` – Prints mail header lines to show you from whom your mail originated

SYNOPSIS

`/usr/ucb/from` [`-s` *sender*] [*user*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `from` utility prints out the mail header lines in your mailbox file to show you from whom your mail originated.

The `from` utility accepts the following option and operand:

`-s` *sender* Prints only headers for mail sent by *sender*.

user Examines *user*'s mailbox instead of your own. To examine another user's mailbox, you must have the appropriate read privileges.

FILES

`/usr/mail/*` Post office directory

SEE ALSO

`mail`(1), `mailx`(1)

NAME

`fsplit` – Splits Fortran files

SYNOPSIS

`fsplit` [`-s`] [*files*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `fsplit` utility splits the named Fortran *files* into separate files, with one routine per file, and lists them. If *files* is not specified, data is read from standard input. A procedure includes the following program segments: `blockdata`, `function`, `main program`, and `subroutine`.

The following naming conventions apply:

- The `program` segment is put in file `MAIN.f`.
- The segment *X* is put in file *X*`.f`.
- Duplicate segments *X* are put into file *XN*`.f`.
- Unnamed `blockdata` segments are put into files `blockdata`*N*`.f`.
- Unnamed segments or segments whose name is unable to be determined are placed in file `zzzzzz`*N*`.f`.

N is a unique integer value for each duplicate file which exists.

The `fsplit` utility accepts the following option and operand:

`-s` Strips trailing blanks.

files Names of files to be split.

NOTES

Expand tabs before splitting files.

As documented on this page, the `.f` suffix is used to name the files generated by `fsplit`. If a # character appears in column one of an input file, the suffix used to name the output file will be `.F`, indicating that the preprocessor must be executed on the output file.

BUGS

The `fsplit` utility is a simple program that can be used effectively. However, the full rules of Fortran input make it difficult to parse for anything less than the full input phase of the compiler. The program has been enhanced to permit comments to precede the `subroutine` statement, to permit spaces inside the `end` keyword, to permit `!` -comments following the `end` keyword, and to permit line numbers beginning in column 73 of the `subroutine` statement. However, a Fortran continue statement with the `subroutine` keyword appearing on a different line than the subroutine name causes the file to be named `zzzzzz`*N*`.f`.

The operation of `fsplit` may be verified using `cat`(1) to send the output files to `wc`(1) and comparing the result to the size of the input file to `fsplit`.

FILES

file`*.[fF]`	Fortran source code files of a named program segment
`zzzzzz*.[fF]`	Source code files containing an unnamed program segment
`blockdata*.[fF]`	Source code files of a blockdata segment

SEE ALSO

`cat`(1), `csplit`(1), `expand`(1), `split`(1), `wc`(1)

NAME

ftp – Transfers files to and from a remote network site

SYNOPSIS

/usr/ucb/ftp [-c *copybufsize*] [-d] [-g] [-i] [-n] [-s *sockbufsize*] [-t] [-v] [-Sc *tos*] [-Sd *tos*] [*host* [*port*]]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The ftp utility is the user interface to the Internet standard file transfer protocol (FTP). The program allows users to transfer files to and from a remote network site.

The client host with which ftp will communicate can be specified on the command line. If this is done, ftp immediately attempts to establish a connection to an FTP server on that host; otherwise, ftp enters its command interpreter and waits for instructions from the user.
When ftp is waiting for commands from the user, the prompt ftp> is provided for the user. If the user supplies insufficient command arguments, ftp prompts for them.

The port number of the ftpd daemon may be specified if the host name is specified. If you do not specify a port number, the default ftp port from the /etc/services file is used.

You can interrupt the ftp utility by pressing the terminal interrupt key, usually <CONTROL-c>. If this is done while ftp is attempting to carry out a command, ftp reverts to the command interpreter and displays the ftp> prompt; otherwise, ftp is terminated.

If your site has installed Kerberos version 4, ftp will attempt to perform Kerberos authentication. If authentication negotiation is successful, an encoded authentication string is exchanged between ftp and ftpd. This version of ftp protects authentication data using base 64 encoding. When authentication is successful, all commands sent over the control channel are protected with cryptographic checksum or encryption. Data sent over the data channel may be protected with cryptographic checksum or encryption. Encryption is available only in the United States and Canada.

The ftp utility accepts the following options:

-c *copybufsize*
: Sets the copy buffer size. A numeric argument sets the buffer to that size. The letter K or k can follow a numeric buffer size to specify a multiple of 1024.

-d
: Enables debugging.

-g
: Disables file name globbing (see the glob option).

-i
: Turns off interactive prompting during multiple file transfers.

`-n` Restrains `ftp` from attempting autologin during initial connection. If autologin is enabled, `ftp` checks the `.netrc` file in the user's home directory for an entry that describes an account on the remote machine. If no entry exists, `ftp` uses the login name on the local machine as the user identity on the remote machine, and prompts for a password and, optionally, an account with which to log in.

`-s` *sockbufsize*
Sets the socket buffer size. A numeric argument sets the buffer to that size. The letter `K` or `k` can follow a numeric buffer size to specify a multiple of 1024.

`-t` Enables tracing.

`-v` Forces `ftp` to show all responses from the remote server, and reports data transfer statistics (verbose option).

`-Sc` *tos* Sets the Internet Protocol (IP) type of service option for the FTP control connection to the value *tos*, which may be a numeric Type-of-Service (TOS) value or a symbolic TOS name that is found in the `/etc/iptos` file.

`-Sd` *tos* Sets the IP type of service option for the FTP data connection to the value *tos*, which may be a numeric TOS value or a symbolic TOS name that is found in the `/etc/iptos` file.

host [*port*] Specifies the host or port with which `ftp` will communicate.

The `ftp` utility recognizes the following commands. They may be given aliases but they must remain unique.

`!` [*command* [*args*]]
Invokes an interactive shell on the local machine. If arguments exist, the first is considered a command to execute directly, and the others are considered arguments.

`$` *macro-name* [*args*]
Executes the macro *macro-name* that is defined by the `macdef` command. Arguments are passed to the macro unglobbed.

`account` [*passwd*]
Supplies a supplemental password that is required by a remote system to access resources after a login is completed successfully. If no argument is included, the user is prompted for an account password through a nonechoing input mode.

`allo` [*arg*]
Specifies whether the `FTP ALLO` and `FTP SIZE` commands will be sent before `put` and `get` commands are executed. The `FTP ALLO` command informs the server of the size of the file that will be sent by using the `put` command so that it can preallocate the disk space if necessary. The `FTP SIZE` command inquires about the size of a file that will be retrieved by using the `get`(1) command, so that the disk space can be preallocated before the file is fetched.

If *arg* is `on`, the `FTP ALLO` or `FTP SIZE` command is sent automatically before a `put` or `get` is executed; if *arg* is `off`, they are not sent. The default is `on`.

`append` *local-file* [*remote-file*]
: Appends *local-file* to a file on the remote machine. If *remote-file* is unspecified, the name of *local-file* is used to name *remote-file*. File transfer uses the current settings for `type`, `format`, `mode`, and `structure`.

`ascii` Sets the file transfer type to network ASCII. This is the default type.

`bell` Sounds a bell after each file transfer command is completed.

`binary` Sets the file transfer type to support binary image transfer.

`bye` Terminates the FTP session with the remote server and exits `ftp`.

`case` Toggles remote computer file name case mapping during `mget` commands. When `case` is `on` (default is `off`), remote computer file names that leave all letters in uppercase are written in the local directory with the letters mapped to lowercase.

`cd` *remote-directory*
: Changes the working directory on the remote machine to *remote-directory*.

`cdup` Changes the remote machine working directory to the parent of the current remote machine working directory.

`chmod` *remote-file*
: Changes file permissions of a remote file.

`clear` Sets the file protection level to clear text for file transfer. All commands sent over the control channel are protected by cryptographic checksum until a different protection level is specified.

`close` Terminates the FTP session with the remote server and returns to the command-line interpreter. Erases any defined macros.

`copybuf` [*arg*]
: Sets the copy buffer size. This buffer is used for reading and writing between the data socket and the source or destination file. FTP automatically chooses a copy buffer size; however, you can alter the selection. `copybuf` takes an optional argument. An argument of `off` turns off copy buffer sizing and the buffer defaults to the size specified in the `tcp_config.h` file by the `COPYBUFSIZE` variable. An argument of `auto` returns buffer sizing to automatic. A numeric argument sets the buffer to that size. The letter `K` or `k` can follow a numeric buffer size to specify a multiple of 1024. Without an argument, `copybuf` shows the current copy buffering. The default setting is `auto`.

`cr` Toggles carriage-return stripping during ASCII type file retrieval. Denotes records by a carriage return or linefeed sequence during ASCII type file transfer. When `cr` is `on` (the default), carriage returns are stripped from this sequence to conform with the UNIX single-line-feed record delimiter. Records on remote systems other than UNIX can contain single linefeeds; when an ASCII type transfer is made, you can distinguish these linefeeds from a record delimiter only when `cr` is `off`.

`delete` *remote-file*
: Deletes the file *remote-file* on the remote machine.

`debug` [*debug-value*]
: Toggles debugging mode. If an optional *debug-value* is specified, it is used to set the debugging level. When debugging is on, `ftp` prints each command that is sent to the remote machine when it is preceded by the string `-->`.

`dir` [*remote-directory*] [*local-file*]
: Prints a listing of the contents of *remote-directory* and, optionally, places the output in *local-file*. If no directory is specified, the current working directory on the remote machine is used. If *local-file* is not specified, or *local-file* is `-`, output goes to the terminal. The `dir` command is synonymous with `ls`.

`disconnect`
: Indicates a synonym for `close`.

`form` *format*
: Sets the file transfer format to *format*. The default format is nonprint. Currently, only the default is supported.

`fullbuf` Toggles the use of full buffers on write-to-disk functions. During a get operation, FTP fills the copy buffer with reads from the data socket before writing the contents of the buffer to the destination file. You can use the `fullbuf` command to disable this feature.

`get` *remote-file* [*local-file*]
: Retrieves *remote-file* and stores it on the local machine. If the name of *local-file* is not specified, it is given the same name that it has on the remote machine; however, it can be altered by the current `case`, `ntrans`, and `nmap` settings. The current settings for `type`, `form`, `mode`, and `structure` are used when the file is transferred.

`glob` Toggles file name expansion for `mdelete`, `mget`, and `mput`. If globbing is turned off with the `glob` command, the file name arguments are taken literally and not expanded. Globbing for `mput` is done the same as for `csh`(1). For `mdelete` and `mget`, each remote file name is expanded separately on the remote machine and the lists are not merged. Expansion of a directory name is probably different from expansion of the name of an ordinary file. The exact result depends on the foreign operating system and `ftp` server, and it can be previewed by using `mls` *remote-files* `-`. The `mget` and `mput` commands are not used for transferring entire directory subtrees of files. To do that, transfer a `tar`(1) archive of the subtree (in binary mode).

`hash` Toggles the printing of a hash sign (#) on each data block that is transferred. The size of a data block is 4096 bytes.

`help` [*command*]
: Prints an informative message about the meaning of *command*. If no argument is given, `ftp` prints a list of the known commands.

`idle` [*seconds*]
: Gets or sets idle timer on the remote side.

`image` Indicates a synonym for binary.

`lcd` [*directory*]
Changes the working directory on the local machine. If *directory* is not specified, the user's home directory is used.

`ls` [*remote-directory*] [*local-file*]
Prints an abbreviated listing of the contents of a directory on the remote machine. If *remote-directory* is unspecified, the current working directory is used. If *local-file* is not specified, or if *local-file* is `-`, the output is sent to the terminal.

`macdef` *macro-name*
Defines a macro. Subsequent lines are stored as the macro *macro-name*; a null line (consecutive newline characters in a file or carriage returns from the terminal) terminates the macro input mode. All defined macros leave a limit of 16 macros and 4096 total characters. Macros remain defined until a `close` command is executed. The macro processor interprets `$` and `\` as special characters. A `$` followed by a number is replaced by the corresponding argument on the macro invocation command line. A `$` followed by an `i` signals the macro processor that the executing macro must be looped. On the first pass, `$i` is replaced by the first argument on the macro invocation command line; on the second pass, it is replaced by the second argument, and so on. A `\` followed by any character is replaced by that character. Use the `\` to prevent special treatment of the `$`.

`mdelete` *remote-files*
Deletes the specified files on the remote machine. If globbing is enabled, you can use `ls` to expand the specification of *remote-files* first.

`mdir` *remote-files local-file*
Provides a function similar to `dir`, except that multiple remote files can be specified. If interactive prompting is on, `ftp` prompts the user to verify that the last argument is the target local file that receives the `mdir` output.

`mget` *remote-files*
Expands the *remote-files* on the remote machine and performs a `get` process for each file name that is produced. See `glob` for details of the file name expansion. The resulting file names are then processed according to `case`, `ntrans`, and `nmap` settings. Files are transferred into the local working directory, which can be changed by entering `lcd` *directory*; new local directories can be created by entering `! mkdir` *directory*.

`mkdir` *directory-name*
Makes a directory on the remote machine.

`mls` *remote-files local-file*
Provides a function similar to `ls`, except multiple remote files can be specified. If interactive prompting is on, `ftp` prompts the user to verify that the last argument is the target local file that receives the `mls` output.

`mode` [*mode-name*]
Sets the file transfer mode to *mode-name.* The default mode is stream mode. (Currently, only the default is supported.)

`modtime` *remote-file*
Shows last modification time of a remote file.

`mput` *local-files*
Expands wildcards on the list of local files that are given as arguments and performs a `put` operation for each file in the resulting list. See `glob` for details of file name expansion. Resulting file names are then processed according to `ntrans` and `nmap` settings.

`newer` *remote-file* [*local-file*]
Gets file if the remote file is newer than the local file.

`nlist` [*remote-directory* [*local-file*]]
Lists the contents of the remote directory.

`nmap` [*inpattern outpattern*]
Sets or unsets the file name-mapping mechanism. If no arguments are specified, the file name-mapping mechanism is unset. If arguments are specified, remote file names are mapped during the execution of `mput` commands and `put` commands that are issued without a specified remote target file name; local file names are mapped during the execution of `mget` commands and `get` commands that are issued without a specified local target file name.

This command is useful when you are connecting to a remote system other than UNIX that uses different file naming conventions or practices. The mapping follows the pattern set by *inpattern* and *outpattern*; *inpattern* is a template for incoming file names (which can be already processed according to the `ntrans` and `case` settings).

To template variables, include the sequences `$1`, `$2`, ..., `$9` in *inpattern*. Use `\` to prevent this special treatment of the `$` character. All other characters are treated literally, and they are used to determine the `nmap` *inpattern* variable values (for example, given *inpattern* `$1.$2` and the remote file name `mydata.data`, `$1` has the value `mydata`, and `$2` has the value `data`). The *outpattern* determines the resulting mapped file name. The sequences `$1`, `$2`, ..., `$9` are replaced by any value that results from the *inpattern* template. The sequence `$0` is replaced by the original file name. If *seq1* is not a null string, the sequence [*seq1*,*seq2*] is replaced by *seq1*; otherwise, it is replaced by *seq2* (for example, `nmap $1.$2.$3 [$1,$2].[$2,file]` yields the output file name `myfile.data` for input file names `myfile.data` and `myfile.data.old`, and it yields `myfile.file` for the input file name `myfile`, and yields `myfile.myfile` for the input file name `.myfile`). You can include spaces in *outpattern* (for example, `nmap $1.$2.$3 [$1,$2] [$2,file]` yields the same results as the previous example, except that the output file names will have a space in place of the period (.)). Use the `\` character to prevent special treatment of the `$`, `[`, `]`, and `,` characters.

`ntrans` [*inchars* [*outchars*]]

Sets or unsets the file name, character-translation mechanism. If no arguments are specified, the file name, character-translation mechanism is unset. If arguments are specified, characters in remote file names are translated during the execution of `mput` commands and `put` commands that are issued without a specified remote target file name; characters in local file names are translated during execution of `mget` commands and `get` commands that are issued without a specified local target file name. This command is useful when you are connecting to a remote system other than UNIX that uses different file naming conventions or practices. A character in a file name that matches a character in *inchars* is replaced by the corresponding character in *outchars*. If the character's position in *inchars* is longer than the length of *outchars*, the character is deleted from the file name.

`open` *host* [*port*]

Establishes a connection to the specified *host* FTP server. If an optional *port* number is supplied, `ftp` attempts to contact an FTP server at that port. If autologin is enabled (default), `ftp` also attempts to log in the user to the FTP server automatically.

`private` Sets the file protection level to `private` for the control channel and file transfer. This command is not available outside of the United States and Canada. All messages sent on the control and data channels are protected by encryption.

`protect` [`clear` | `safe` | `private`]

Sets the protection level for file transfer. The `private` option is not available outside of the United States and Canada.

`prompt` Toggles interactive prompting. Interactive prompting occurs during multiple file transfers to allow users to retrieve or store files selectively. If prompting is turned off, an `mget` or `mput` command transfers all files. The default for `prompt` is `on`. If started with the `-i` option, `prompt` starts set at `off`.

`proxy` *ftp-command*

Executes an `ftp` utility on a secondary control connection. This command allows simultaneous connection for transferring files between servers to two remote FTP servers. The first `proxy` command must be open to establish the secondary control connection. Enter the `proxy ?` command to see other `ftp` utilities that are executable on the secondary connection. The following commands act as follows when prefaced with `proxy`:

- `open` does not define new macros during the autologin process.
- `close` does not erase existing macro definitions.
- `get` and `mget` transfer files from the host on the primary control connection to the host on the secondary control connection.

- `put`, `mput`, and `append` transfer files from the host on the secondary control connection to the host on the primary control connection.

  Third-party file transfers depend on support of the `ftp` protocol `PASV` command by the server on the secondary control connection.

  Proxy file transfers are not supported for safe and private protection modes.

`put` *local-file* [*remote-file*]
: Stores *local-file* on the remote machine. If *remote-file* is unspecified, the name of *local-file* is used after processing according to any `ntrans` and `nmap` setting in naming *remote-file*. File transfer uses the current settings for `type`, `format`, `mode`, and `structure`.

`pwd`
: Prints the name of the current working directory on the remote machine.

`quit`
: Indicates a synonym for `bye`.

`quote` *arguments*
: The arguments specified are sent to the remote FTP server exactly as specified. In return, one FTP reply code is expected.

`rawbuf`
: Toggles the use of raw I/O on write-to-disk and read-from-disk functions. Usually, the system overhead is lower when raw I/O is used. The default is that raw I/O is enabled.

`recv` *remote-file* [*local-file*]
: Indicates a synonym for `get`.

`rename` [*from*] [*to*]
: Renames the file *from* on the remote machine to the file *to*.

`reget` *remote-file* [*local-file*]
: Gets file restarting at the end of the local file.

`reset`
: Clears the reply queue. This command resynchronizes the command and reply sequencing by using the remote FTP server. Resynchronization may be necessary when the remote server violates the FTP protocol.

`restart` *bytecount*
: Restarts the file transfer at *bytecount*.

`rhelp` *command-name*
: Requests help from the remote `ftp` server. If *command-name* is specified, it is also supplied to the server.

`rmdir` *directory-name*
: Deletes a directory on the remote machine.

`rstatus`
: Shows the status of the remote machine.

`runique` Toggles storing of files that have unique file names on the local system. If a file already exists with a name equal to the target local file name for a `get` or `mget` command, a `.1` is appended to the name. If the resulting name matches another existing file, a `.2` is appended to the original name. If this process continues up to `.99`, an error message is printed, and the transfer does not occur. The generated unique file name is reported. `runique` does not affect local files that are generated from a shell command. The default value is `off`.

`safe` Sets the file protection level to `safe` for the control channel and file transfer. All messages sent on the control and data channels are protected by cryptographic checksum.

`send` *local-file* [*remote-file*]
Indicates a synonym for `put`.

`sendport`
Toggles the use of `PORT` commands. By default, `ftp` attempts to use a `PORT` command when establishing a connection for each data transfer. The use of `PORT` commands can prevent delays when performing multiple file transfers. If the `PORT` command fails, `ftp` uses the default data port. When the use of `PORT` commands is disabled, no attempt is made to use `PORT` commands for each data transfer. This is useful for certain FTP implementations that ignore `PORT` commands, but incorrectly indicate that they are accepted.

`showbuf` Toggles display of copy buffer and socket buffer sizing information during a transfer. Default is no display of information.

`site` *arguments*
The specified arguments are sent to the remote FTP server as arguments to an FTP `SITE` command. In return, one FTP reply code is expected.

`size` *remote-file*
Shows the size of the remote file.

`sockbuf` [*arg*]
Sets the socket buffer size. This kernel buffer is used for data transfer on the data socket. FTP automatically chooses a socket buffer size; however, you can alter the selection. `sockbuf` takes an optional argument. An argument of `off` turns off socket buffer sizing and the buffer defaults to the kernel's default socket buffer size. An argument of `auto` returns buffer sizing to automatic. A numeric argument sets the buffer to that size. The letter `K` or `k` can follow a numeric buffer size to specify a multiple of 1024. Without an argument, `sockbuf` shows the current socket buffering. The default setting is `auto`.

`status` Shows the current status of `ftp`.

`struct` [*struct-name*]
Sets the file transfer structure to *struct-name*. By default, file structure is used. Currently, only the default is supported.

`sunique` Toggles storing of files that have unique file names on a remote machine. The remote FTP server must support the FTP protocol `STOU` command. The remote server reports unique names. The default value is `off`.

`system` Shows the remote system type.

`tenex` Sets the file transfer *type* to that which is needed to talk to TENEX machines.

`trace` Toggles packet tracing.

`type` [*type-name*]
Sets the file transfer `type` to *type-name*. If *type-name* is not specified, the current type is printed. The three valid types are `tenex`, `binary`, and `ascii`, which is the default.

`umask` *mask*
Gets and sets `umask` on the remote side.

`user` *user-name* [*password*] [*account*]
Identifies you to the remote FTP server. If *password* is not specified and the server requires it, `ftp` prompts you for it (after disabling the local echo). If *account* is not specified, and the FTP server requires it, you are prompted for it. If an account field is specified, an account command is relayed to the remote server after the login sequence is completed if the remote server does not require it to log on. Unless you invoke `ftp` with the autologin disabled, this process is completed automatically during the initial connection to the FTP server.

`verbose` Toggles verbose mode. In verbose mode, all responses from the FTP server are displayed to the user. If `verbose` is on or when a file transfer completes, statistics are reported about the efficiency of the transfer. If `ftp` is started with the `-v` option or if input is a terminal, the default is `on`.

`winshift` [*value*]
Gets or sets the value for the TCP window shift option to be used on data connections. If no argument is specified, this option reports the current value. If *value* is `off`, the option is disabled; if *value* is `on`, the option is enabled with a value of 4; if *value* is an integer value between 0 and 14, the TCP window shift option is enabled with that value. Because the client side of an FTP always performs a passive open operation, you do not have to disable the sending of the TCP window shift option (if the incoming SYN packet does not contain the option, none will be sent in the SYN,ACK packet, regardless of whether the application has enabled the option).

`?` [*command*]
Indicates a synonym for help.

Command arguments that have embedded spaces may be enclosed with quotation marks.

Aborting a File Transfer

To abort a file transfer, use the terminal interrupt key (`<CONTROL-c>`). Sending transfers is halted immediately. Receiving transfers is halted by sending a FTP protocol `ABOR` command to the remote server or by discarding any further data that is received. The speed at which this is accomplished depends on the remote server's support for `ABOR` processing. If the remote server does not support the `ABOR` command, an `ftp>` prompt does not appear until the remote server completes sending the requested file.

The terminal interrupt key sequence is ignored when ftp completes any local processing and is waiting for a reply from the remote server. A long delay in this mode can result from the ABOR processing described in the preceding paragraph, or from the unexpected remote server behavior, including violations of the ftp protocol. If the delay is caused by unexpected remote server behavior, the local ftp program must be killed by hand.

File Naming Conventions

Files specified as arguments to ftp utilities are processed according to the following rules:

- If the file name is specified as a dash (-), stdin (for reading) or stdout (for writing) is used.
- If the first character of the file name is |, the remainder of the argument is interpreted as a shell command. ftp then forks a shell with the argument supplied and reads (or writes) from stdout (or stdin). If the shell command includes spaces, the argument must be enclosed in quotation marks (for example, "| ls -lt"). A particularly useful example of this mechanism is dir *directory_name* "|pg".
- Failing the preceding checks, if globbing is enabled, local file names are expanded according to the rules used in csh(1); see the glob command. If the ftp utility expects a single local file (for example, put), only the first file name that glob operation generates is used.
- When using mget and get commands that have unspecified local file names, the local file name is the remote file name, which can be altered by a case, ntrans, or nmap setting. When runique is on, the resulting file name may then be altered.
- When using mput and put commands that have unspecified remote file names, the remote file name is the local file name, which may be altered by an ntrans or nmap setting. When sunique is on, the remote server can alter the resulting file name.

File Transfer Parameters

The ftp specification specifies many parameters that can affect a file transfer. The type can be one of ASCII, image (binary), EBCDIC, and local byte size (for PDP-10s and PDP-20s). The ftp utility supports the ASCII and image types of file transfer, and also local byte size 8 for TENEX mode transfers.

The ftp utility supports only the default values for the remaining file transfer parameters: mode, form, and struct.

NOTES

The SHELL environment variable defines the shell that is used in shell-escapes. If SHELL is undefined, the default shell /bin/sh is used. The HOME environment variable is used to locate the user's home directory when it reads the .netrc file.

BUGS

Several FTP server implementations do not support operations such as `pwd`.

You must use the `mget` and `mdelete` commands with caution. If you specify a directory that expects a plain file name, unexpected results can be produced.

EXAMPLES

The following example transmits an executable binary file from the system `host` to the system `cray`. Typewriter bold font indicates user input:

```
host% ftp cray
Cray FTP server ready.
Name (cray:dep): dep
Password (cray:dep):            Enter your password
Password required for dep.
User dep logged in.
ftp> binary
Type set to I.
ftp> put a.out
PORT command successful.
Opening data connection for a.out (port number).
Transfer complete.
N bytes sent in n seconds
ftp> quit
Goodbye.
```

FILES

`/etc/hosts`	File that contains the database of all locally known hosts on the TCP/IP network
`$HOME/.netrc`	File that contains login information required for `ftp` access to a remote machine

SEE ALSO

`glob`(3C) in the *UNICOS System Libraries Reference Manual*, Cray Research publication SR–2080

`ftpusers`(5), `netrc`(5) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

`fta`(8), `ftpd`(8) in the *UNICOS Administrator Commands Reference Manual*, Cray Research publication SR–2022

Software Overview for Users, Cray Research publication SG–2052

NAME

fts – Performs file transfer server function for bftp(1B)

SYNOPSIS

/usr/ucb/fts

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The fts utility is the transfer request server for the background file transfer program (BFTP). You can use bftp(1B) to submit a request to have a file transferred at some time in the future by the use of the standard Internet file transfer protocol (FTP), which is described in RFC 959.

The fts utility is not a user command and is used only in association with bftp(1B).

For information on BFTP, see RFC 1068.

SEE ALSO

at(1), bftp(1B), crontab(1), ftp(1B)

cron(8), ftpd(8) in the *UNICOS Administrator Commands Reference Manual*, Cray Research publication SR–2022

RFC 959, Postel, J.B., Reynolds, J.K. File Transfer Protocol. October 1985: 69 pages.
RFC 1068, DeSchon, A.L., Braden, R.T. Background File Transfer Program (BFTP). August 1988; 27 pages.

NAME

gencat – Generates a message catalog

SYNOPSIS

gencat *catfile msgfiles*

IMPLEMENTATION

All Cray Research systems

STANDARDS

XPG4

DESCRIPTION

This utility was formerly called catgen(8). It has been renamed gencat(1) and moved to /usr/bin for UNICOS 8.3, UNICOS 9.0, and subsequent releases to conform with the X/Open XPG4 specification. The utility that was called gencat(8) in former releases is now called mfsck(8). The functionality of neither utility has changed, only the names.

The gencat utility generates a formatted message catalog from the output of the caterr(1) utility. The files output from caterr are input to gencat *msgfile*.

The gencat utility accepts the following arguments:

catfile Specifies the name to be given to the catalog file output from gencat. If *catfile* exists, its messages are included in the updated *catfile*. Messages and explanations from the input files replace messages and explanations of identical set and message numbers in the existing *catfile*. If *catfile* does not exist, gencat creates it.

msgfiles Specifies the name of the file or files input to gencat. The input to gencat must be a file output from caterr(1). The caterr utility processes the message file; the gencat utility outputs the message or explanation catalog.

NOTES

When the -c option is specified on the caterr command line, gencat is called from caterr. It is recommended that you call gencat in this fashion. The gencat utility exists as a separate utility to maintain compatibility with industry standards for message processing. No advantage exists in calling gencat directly.

Message catalogs that gencat produces are binary encoded. This means that their portability cannot be guaranteed among various types of machines. Therefore, just as C programs must be recompiled for each type of machine, message catalogs must be re-created using gencat.

EXIT STATUS

An exit value of 0 is returned if the command completed successfully. An exit value greater than 0 is returned if an error occurred that prevented successful completion.

EXAMPLES

The following example updates the existing catalog `dbg.cat` with the information in the `dbg.in` file:

```
gencat dbg.cat dbg.in
```

SEE ALSO

`caterr`(1), `catxt`(1), `explain`(1), `whichcat`(1)

`catgetmsg`(3C), `catgets`(3C), `catmsgfmt`(3C), `catopen`(3C) in the *UNICOS System Libraries Reference Manual*, Cray Research publication SR–2080

`nl_types`(5) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

Cray Message System Programmer's Guide, Cray Research publication SG–2121

NAME

get – Gets a version of an SCCS file

SYNOPSIS

get [-a "*seq-no*"] [-b] [-c *cutoff*] [-e] [-g] [-i *list*] [-k] [-l] [-L] [-m] [-n] [-p] [-r *SID*] [-s] [-t] [-w *string*] [-x *list*] *file*

Obsolescent version; may not be supported in future releases:

get [-a "*seq-no*"] [-b] [-c *cutoff*] [-e] [-g] [-i *list*] [-k] [-l[p]] [-m] [-n] [-p] [-r *SID*] [-s] [-t] [-w *string*] [-x *list*] *file*

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The get utility generates an ASCII text file from each named Source Code Control System (SCCS) file according to the specifications given by its options, which begin with a dash (-). The options may be specified in any order, but all options apply to all named SCCS files. If a directory is named, get behaves as though each file in the directory were specified as a named file, except that non-SCCS files (last component of the path name does not begin with s.) and unreadable files are silently ignored. If a name of - is given, the standard input is read; each line of the standard input is taken to be the name of an SCCS file to be processed. Again, non-SCCS files and unreadable files are silently ignored.

The generated text is usually written into a file called the *g-file* whose name is derived from the SCCS file name by simply removing the leading s. (See the FILES section.)

Each of the options is explained below as though only one SCCS file is to be processed, but the effects of any option applies independently to each named file.

-a"*seq-no*" The delta sequence number of the SCCS file delta (version) to be retrieved (see sccsfile(5)). This option is used by the comb(1) command; it is not a generally useful option, and users should not use it. If both the -r and -a options are specified, the -a option is used. Care should be taken when using the -a option in conjunction with the -e option, because the source identifier (SID) of the delta to be created may not be what you expect. The -r option can be used with the -a and -e options to control the naming of the SID for the delta to be created.

`-b` Used with the `-e` option to indicate that the new delta should have an SID in a new branch as shown in the table below. This option is ignored if the `b` flag is not present in the file (see `admin`(1)) or if the retrieved *delta* is not a leaf *delta*. (A leaf *delta* is one that has no successors on the SCCS file tree.)

Note: A branch *delta* may always be created from a nonleaf *delta*.

`-c`*cutoff* Indicates the *cutoff* date-time. This appears in the form:

YY[*MM*[*DD*[*HH*[*MM*[*SS*]]]]]

No changes (deltas) to the SCCS file that were created after the specified *cutoff* date-time are included in the generated ASCII text file. Units omitted from the date-time default to their maximum possible values; that is, `-c7502` is equivalent to `-c750228235959`. Any number of nonnumeric characters may separate the various 2-digit pieces of the *cutoff* date-time. This feature lets you specify a *cutoff* date-time in the form `-c77/2/2 9:22:25`. This implies that you may use the `%E%` and `%U%` identification keywords (see below) for nested `gets`:

```
~!get "-c%E% %U%" s.file
```

`-e` Indicates that the `get` is for the purpose of editing or making a change (delta) to the SCCS file through a subsequent use of `delta`(1). The `-e` option used in a `get` for a particular version (SID) of the SCCS file prevents further `get`s for editing on the same SID until *delta* is executed or the `j` (joint edit) flag is set in the SCCS file (see `admin`(1)). Concurrent use of `get -e` for different SIDs is always allowed.

If the *g-file* generated by `get` with an `-e` option is accidentally damaged in the process of editing it, it may be regenerated by re-executing the `get` command with the `-k` option in place of the `-e` option.

SCCS file protection specified by using the ceiling, floor, and authorized user list stored in the SCCS file (see `admin`(1)) are enforced when the `-e` option is used.

`-g` Suppresses the actual retrieval of text from the SCCS file. It is primarily used to generate an *l-file*, or to verify the existence of a particular SID.

`-i`*list* A *list* of deltas to be included (forced to be applied) in the creation of the generated file. The *list* has the following syntax:

<list> ::= <range> | <list> , <range>
<range> ::= SID | SID - SID

SID, the SCCS identification of a delta, may be in any form shown in the "SID specified" column of the table below. Partial SIDs are interpreted as shown in the "SID retrieved" column of the table below.

`-k` Suppresses replacement of identification keywords in the retrieved text by their value. The `-k` option is implied by the `-e` option.

`-l` Causes a delta summary to be written into an *l-file*. See the FILES section for the format of the *l-file*.

`-L` Causes a delta summary to be written to standard output.

`-lp` Equivalent to `-L`.

`-m` Causes each text line retrieved from the SCCS file to be preceded by the SID of the delta that inserted the text line in the SCCS file. The format is: SID, followed by a horizontal tab, followed by the text line.

`-n` Causes each generated text line to be preceded with the `%M%` identification keyword value. The format is: `%M%` value, followed by a horizontal tab, followed by the text line. When both the `-m` and `-n` options are used, the format is: `%M%` value, followed by a horizontal tab, followed by the `-m` option generated format.

`-p` Causes the text retrieved from the SCCS file to be written on the standard output. No *g-file* is created. All output that normally goes to the standard output goes to file descriptor 2 instead, unless the `-s` option is used, in which case it disappears.

`-r` *SID* The *S*CCS *ID*entification string (SID) of the version (delta) of an SCCS file to be retrieved. Table 1 shows, for the most useful cases, what version of an SCCS file is retrieved (as well as the SID of the version to be created eventually by `delta`(1) if the `-e` option is also used), as a function of the SID specified.

`-s` Suppresses all output usually written on the standard output. However, fatal error messages (which always go to file descriptor 2) remain unaffected.

`-t` Used to access the most recently created (‘‘top’’) delta in a given release (such as, `-r1`), or release and level (such as, `-r1.2`).

`-w` *string* Substitutes *string* for all occurrences of "`% w%`" when using the `get` command on a file.

`-x` *list* A *list* of deltas to be excluded (forced not to be applied) in the creation of the generated file. See the `-i` option for the *list* format.

file Specifies the SCCS file to get.

For each file processed, `get` responds (on the standard output) with the SID being accessed and with the number of lines retrieved from the SCCS file.

If the `-e` option is used, the SID of the delta to be made appears after the SID accessed and before the number of lines generated. If there is more than one named file or if a directory or standard input is named, each file name is printed (preceded by a new line) before it is processed. If the `-i` option is used, included deltas are listed following the notation `Included`; if the `-x` option is used, excluded deltas are listed following the notation `Excluded`.

SID† specified	-b keyletter used††	Other conditions	SID retrieved	SID of delta to be created
None†††	No	R defaults to mR	mR.mL	mR.(mL+1)
None†††	Yes	R defaults to mR	mR.mL	mR.mL.(mB+1).1
R	No	R > mR	mR.mL	R.1††††
R	No	R = mR	mR.mL	mR.(mL+1)
R	Yes	R > mR	mR.mL	mR.mL.(mB+1).1
R	Yes	R = mR	mR.mL	mR.mL.(mB+1).1
R	–	R < mR and R does *not* exist	hR.mL†††††	hR.mL.(mB+1).1
R	–	Trunk successor in release > R and R exists	R.mL	R.mL.(mB+1).1
R.L	No	No trunk successor	R.L	R.(L+1)
R.L	Yes	No trunk successor	R.L	R.L.(mB+1).1
R.L	–	Trunk successor in release ≥ R	R.L	R.L.(mB+1).1
R.L.B	No	No branch successor	R.L.B.mS	R.L.B.(mS+1)
R.L.B	Yes	No branch successor	R.L.B.mS	R.L.(mB+1).1
R.L.B.S	No	No branch successor	R.L.B.S	R.L.B.(S+1)
R.L.B.S	Yes	No branch successor	R.L.B.S	R.L.(mB+1).1
R.L.B.S	–	Branch successor	R.L.B.S	R.L.(mB+1).1

† R, L, B, and S are the release, level, branch, and sequence components of the SID, respectively; m means maximum. Thus, for example, R.mL means the "maximum level number within release R"; R.L.(mB+1).1 means "the first sequence number on the new branch (that is, maximum branch number plus one) of level L within release R." Note that if the SID specified is of the form R.L, R.L.B, or R.L.B.S, each of the specified components must exist.

†† hR is the highest existing release that is lower than the specified, nonexistent, release R.

††† This is used to force creation of the first delta in a new release.

†††† The -b option is effective only if the b flag (see admin(1)) is present in the file. An entry of - means ‘‘irrelevant.’’

††††† This case applies if the d (default SID) flag is not present in the file. If the d flag *is* present in the file, the SID obtained from the d flag is interpreted as if it had been specified on the command line. Thus, one of the other cases in this table applies.

Identification Keywords

Identifying information is inserted into the text retrieved from the SCCS file by replacing identification keywords with their value wherever they occur. The following keywords may be used in the text stored in an SCCS file:

Keyword	Value
%M%	Module name: either the value of the m flag in the file (see admin(1)), or if absent, the name of the SCCS file with the leading s. removed.
%I%	SCCS identification (SID) (%R%.%L%.%B%.%S%) of the retrieved text
%R%	Release.
%L%	Level.
%B%	Branch.
%S%	Sequence.
%D%	Current date (*Y*/*MM*/*DD*).
%H%	Current date (*MM*/*DD*/*YY*).
%T%	Current time (*HH*:*MM*:*SS*).
%E%	Date newest applied delta was created (*YY/MM/DD).*
%G%	Date newest applied delta was created (*MM/DD/YY).*
%U%	Time newest applied delta was created (*HH:MM:SS).*
%Y%	Module type: value of the t flag in the SCCS file (see admin(1)).
%F%	SCCS file name.
%P%	Fully qualified SCCS file name.
%Q%	The value of the q flag in the file (see admin(1)).
%C%	Current line number. This keyword identifies messages output by the program such as ‘‘this should not have happened’’ type errors. It is not intended to be used on every line to provide sequence numbers.
%Z%	The 4-character string @(#) recognizable by what(1).
%W%	A shorthand notation for constructing what(1) strings for UNICOS system program files. %W%~=~%Z%%M%<horizontal-tab>%I%
%A%	Another shorthand notation for constructing what(1) strings for non-UNICOS system program files. %A%~=~%Z%%Y%~%M%~%I%%Z%

EXIT STATUS

The following exit values are returned:

0 Successful completion.

>0 An error occurred.

MESSAGES

Use help(1) for explanations.

BUGS

If the effective user has write permission (either explicitly or implicitly) in the directory containing the SCCS files, but the real user does not, only one file may be named when the `-e` option is used.

EXAMPLES

The following example retrieves the latest delta for editing. The file `example.c` is created. User input is indicated with bold type:

```
$ get -e s.example.c
1.1
new delta 1.2
5 lines
$
```

FILES

Several auxiliary files may be created by `get`. These files are known generically as the *g-file*, *l-file*, *p-file*, and *z-file*. The letter before the hyphen is called the tag. An auxiliary file name is formed from the SCCS file name: the last component of all SCCS file names must be of the form `s.`*module-name*, the auxiliary files are named by replacing the leading `s` with the tag. The *g-file* is an exception to this scheme: the *g-file* is named by removing the `s.` prefix. For example, `s.xyz.c`, the auxiliary file names would be `xyz.c`, `l.xyz.c`, `p.xyz.c`, and `z.xyz.c`, respectively.

The *g-file*, which contains the generated text, is created in the current directory (unless the `-p` option is used). A *g-file* is created in all cases, whether or not any lines of text were generated by `get`. It is owned by the real user. If the `-k` option is used or implied, its mode is 644; otherwise, its mode is 444. Only the real user must have write permission in the current directory.

The *l-file* contains a table showing which deltas were applied in generating the retrieved text. The *l-file* is created in the current directory if the `-l` option is used; its mode is 444 and it is owned by the real user. Only the real user needs to have write permission in the current directory.

Lines in the *l-file* have the following format:

- A blank character if the delta was applied; otherwise, `*`.
- A blank character if the delta was applied or was not applied and ignored; `*` if the delta was not applied and was not ignored.
- A code indicating a ‘‘special’’ reason why the delta was or was not applied:

  `I`: Included
  `X`: Excluded
  `C`: Cut off (by a `-c` option)

- Blank
- SCCS identification (SID)

- Tab character
- Date and time (in the form *YY*/*MM*/*DD*˜*HH*:*MM*:*SS*) of creation
- Blank
- Login name of person who created *delta*

The comments and memory resident (MR) data follow on subsequent lines, indented one horizontal tab character. A blank line terminates each entry.

The *p-file* is used to pass information resulting from a `get` with an `-e` option along to *delta*. Its contents are also used to prevent a subsequent execution of `get` with an `-e` option for the same SID until *delta* is executed or the joint edit flag, `j`, (see `admin`(1)) is set in the SCCS file. The *p-file* is created in the directory containing the SCCS file and the effective user must have write permission in that directory. Its mode is 644 and it is owned by the effective user. The format of the *p-file* is: the retrieved SID, followed by a blank, followed by the SID that the new delta will have when it is made, followed by a blank, followed by the login name of the real user, followed by a blank, followed by the date-time the `get` was executed, followed by a blank and the `-i` option if it was present, followed by a blank and the `-x` option if it was present, followed by a new line. There can be an arbitrary number of lines in the *p-file* at any time; no two lines can have the same new delta SID.

The *z-file* serves as a *lock-out* mechanism against simultaneous updates. Its contents are the binary process ID of the command (that is, `get`) that created it. The *z-file* is created in the directory containing the SCCS file for the duration of `get`. The same protection restrictions as those for the *p-file* apply for the *z-file*. The *z-file* is created mode 444.

SEE ALSO

`admin`(1), `cdc`(1), `comb`(1), `delta`(1), `help`(1), `prs`(1), `rmdel`(1), `sact`(1), `sccsdiff`(1), `unget`(1), `val`(1), `vc`(1), `what`(1)

`sccsfile`(5) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

NAME

getconf – Gets configuration values

SYNOPSIS

getconf *system_var*
getconf *path_var pathname*

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

In the first synopsis form, the getconf utility writes to the standard output the value of the variable specified by the *system_var* operand.

In the second synopsis form, the getconf utility writes to the standard output the value of the variable specified by the *path_var* operand for the path specified by the *pathname* operand.

The following operands are supported:

system_var A name of a configuration variable that has a value available from sysconf(2) and confstr(3C). The values in the following table are supported:

ARG_MAX	CRAY_SYSMEM	POSIX2_VERSION
BC_BASE_MAX	CRAY_USRMEM	POSIX_ARG_MAX
BC_DIM_MAX	CS_PATH	POSIX_CHILD_MAX
BC_SCALE_MAX	EXPR_NEST_MAX	POSIX_JOB_CONTROL
BC_STRING_MAX	INT_MAX	POSIX_LINK_MAX
CHARCLASS_NAME_MAX	INT_MIN	POSIX_MAX_CANON
CHAR_BIT	LINE_MAX	POSIX_MAX_INPUT
CHAR_MAX	LONG_BIT	POSIX_NAME_MAX
CHAR_MIN	LONG_MAX	POSIX_NGROUPS_MAX
CHILD_MAX	LONG_MIN	POSIX_OPEN_MAX
CLK_TCK	MB_LEN_MAX	POSIX_PATH_MAX
COLL_WEIGHTS_MAX	MN_NMAX	POSIX_PIPE_BUF
CRAY_AVL	NGROUPS_MAX	POSIX_SAVED_IDS
CRAY_BDM	NL_ARGMAX	POSIX_SSIZE_MAX
CRAY_CHIPSZ	NL_LANGMAX	POSIX_STREAM_MAX
CRAY_CPCYCLE	NL_MSGMAX	POSIX_TZNAME_MAX
CRAY_EMA	NL_SET_MAX	POSIX_VERSION
CRAY_HPM	NL_TEXT_MAX	RE_DUP_MAX

CRAY_IOS	NZERO	SCHAR_MAX
CRAY_MFSUBTYPE	OPEN_MAX	SCHAR_MIN
CRAY_MFTYPE	POSIX2_BC_BASE_MAX	SHRT_MAX
CRAY_NBANKS	POSIX2_BC_DIM_MAX	SHRT_MIN
CRAY_NBUF	POSIX2_BC_SCALE_MAX	SSIZE_MAX
CRAY_NCPU	POSIX2_BC_STRING_MAX	STREAM_MAX
CRAY_NDISK	POSIX2_CHAR_TERM	TMP_MAX
CRAY_NMOUNT	POSIX2_COLL_WEIGHTS_MAX	TZNAME_MAX
CRAY_NPTY	POSIX2_C_BIND	UCHAR_MAX
CRAY_NUSERS	POSIX2_C_DEV	UINT_MAX
CRAY_NVHISP	POSIX2_C_VERSION	ULONG_MAX
CRAY_OS_HZ	POSIX2_EXPR_NEST_MAX	USHRT_MAX
CRAY_RELEASE	POSIX2_FORT_DEV	WORD_BIT
CRAY_SCTRACE	POSIX2_FORT_RUN	XOPEN_VERSION
CRAY_SDS	POSIX2_LINE_MAX	XOPEN_XCU_VERSION
CRAY_SECURE_MAC	POSIX2_LOCALEDEF	XOPEN_XPG2
CRAY_SECURE_SYS	POSIX2_RE_DUP_MAX	XOPEN_XPG3
CRAY_SERIAL	POSIX2_SW_DEV	XOPEN_XPG4
CRAY_SSD	POSIX2_UPE	

The symbol PATH also is recognized, yielding the same value as the confstr(3C) name value CS_PATH.

path_var A name of a configuration variable that has a value available from pathconf(2). The values in the following table are supported:

LINK_MAX	NAME_MAX	POSIX_CHOWN_RESTRICTED
MAX_CANON	PATH_MAX	POSIX_NO_TRUNC
MAX_INPUT	PIPE_BUF	POSIX_VDISABLE

pathname A path name for which the variable specified by *path_var* will be determined.

If the specified variable is valid, but is undefined on the system, getconf writes undefined to stdout. If the variable name is not valid or an error occurs, getconf writes nothing to stdout.

EXIT STATUS

The getconf utility exits with one of the following values:

0 The specified variable is valid, and information about its current state was written successfully.

>0 An error occurred.

EXAMPLES

Example 1: The following command prints out the multigroup size:

```
getconf NGROUPS_MAX
```

Example 2: The following command prints out the value of {NAME_MAX} for a specific directory:

```
getconf NAME_MAX /usr
```

Example 3: The following shell script fragment example shows how to deal more carefully with results that might be unspecified:

```
if value=$(getconf PATH_MAX /usr); then
     if [ "$value" = "undefined" ]; then
          echo PATH_MAX in /usr/ is infinite.
     else
          echo PATH_MAX in /usr is $value.
     fi
else
     echo Error in getconf.
fi
```

SEE ALSO

pathconf(2), sysconf(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

confstr(3C) in the *UNICOS System Libraries Reference Manual*, Cray Research publication SR–2080

NAME

`getopt` – Parses command options

SYNOPSIS

`getopt` *optstring* [*arguments*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `getopt` utility breaks up options in command lines for easy parsing by shell procedures and checks for legal options.

optstring is a string of recognized option letters (see `getopt`(3C)). When a letter is followed by a colon, the option is expected to have an argument that may or may not be separated from it by white space.

arguments is a list of 0 or more blank-separated words. These words are usually options, option-arguments, and operands (nonoption words, such as file names).

The following output from `getopt` is displayed on one line in the order given:

- Each option, optionally followed by a separate word, the option-argument. Each option is preceded by a minus sign.
- The characters `--`. If these characters are not specified, they will be generated automatically by `getopt` to indicate the end of the options list.
- Operands (nonoption words such as file names).

NOTES

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
`system`, `secadm`	In a privileged administrator shell environment, allowed to write shell-redirected output to any file.
`sysadm`	Shell-redirected output is subject to security label restrictions.

If the `PRIV_SU` configuration option is enabled, the super user can write shell-redirected output to any file.

`getopts`(1) is the preferred method for parsing command lines of standard shell procedures.

MESSAGES

When `getopt` encounters an option letter not included in *optstring*, it prints an error message on the standard error.

EXAMPLES

The following standard shell script fragment shows how you might process the arguments for a command that can take the `-a` or `-b` options, as well as the `-o` option, which requires an argument:

```
oarg=
flag=
USAGE="Usage: $0 [-ab] [-o oarg] file1 file2"
opts=‘getopt abo: $*‘
err=$?
set -- $opts
#
if [ $err -eq 0 ]
then
      while [ "$1" != "--" ]
      do
            case $1 in
            -a | -b)    flag=$1;;
            -o)         oarg=$2; shift;;
            esac
            shift
      done
      shift
fi
#
if [ $err -ne 0 -o $# -ne 2 ]
then
      echo $USAGE >&2
      exit 1
.
.
.
```

The following are some of the command lines that both procedures accept as equivalent:

cmd `-abo` *arg file file*
cmd `-ab -o` *arg file file*
cmd `-b -a -o`*arg file file*
cmd `-bo` *arg* `-a` *file file*
cmd `-o` *arg* `-ba --` *file file*
cmd `-a -o` *arg* `-b --` *file file*

Because `-o` takes an argument, the following command line is *not* equivalent to those in the previous list:

cmd `-oab` *arg file file*

SEE ALSO

`getopts`(1), `sh`(1)

`getopt`(3C) in the *UNICOS System Libraries Reference Manual*, Cray Research publication SR–2080

NAME

getopts – Parses utility options

SYNOPSIS

getopts *optstring name* [*args*]

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The getopts utility is used by standard shell procedures to parse positional parameters and to check for legal options. It supports all applicable rules of the utility argument syntax standard.

The *optstring* argument must contain the option letters recognized by the command using getopts; however, when a letter is followed by a colon, the option is expected to have an argument, or group of arguments, which may or may not be separated from it by white space. You should use white space to separate options and their arguments.

Each time it is invoked, getopts places the next option in *name* shell variable and the index of the next argument to be processed in the OPTIND shell variable. Whenever the shell or a shell procedure is invoked, OPTIND is initialized to 1.

When an option requires an option-argument, getopts places it in shell environment variable OPTARG.

If an illegal option is encountered, ? will be placed in *name*.

When the end of options is encountered, getopts exits with a nonzero exit status. You can use the special option ‘‘--’’ to delimit the end of the options.

By default, getopts parses the positional parameters. If you specify extra arguments (*args*) on the getopts command line, getopts will parse them instead.

To ensure that all new shell procedures adhere to the utility argument syntax standard, they should use getopts.

The shell variable specified by the *name* operand, OPTIND, and OPTARG affect the current shell execution environment.

CAUTIONS

Modifying the shell environment variable `OPTIND` to a value other than 1 or parsing different sets of arguments may lead to unexpected results.

NOTES

The `getopts` utility described on this manpage is a built-in utility to the standard shell (`sh`(1)). An executable version of this utility is available in `/usr/bin/getopts`.

MESSAGES

When it encounters an option letter not included in *optstring*, `getopts` prints an error message on standard error. No output to standard error is written in this case if the first character in *optstring* is a colon (`:`).

EXAMPLES

Example 1: The following fragment of a shell program shows how you might process the arguments for a command that can take option `a` or `b`, as well as option `o`, which requires an argument:

```
flag=
oarg=
errflg=0
USAGE="Usage:  util_name [-ab] [-o oarg] file"
#
while getopts abo: option
do
        case $option in
        a | b)          flag=$option;;
        o)              oarg=$OPTARG;;
        \?)            errflg=1;;
        esac
done
shift `expr $OPTIND - 1`
if [ $errflg -ne 0 -o $# -ne 1 ]
then
        echo $USAGE >&2
        exit 2
fi
.
.
.
```

This code accepts any of the following as equivalent:

```
util_name -a -b -o "xxx z yy" file
util_name -a -b -o "xxx z yy" -- file
util_name -ab -o xxx,z,yy file
util_name -ab -o "xxx z yy" file
util_name -o xxx,z,yy -b -a file
```

SEE ALSO

getopt(1), sh(1)

getopt(3C) in the *UNICOS System Libraries Reference Manual*, Cray Research publication SR–2080

NAME

grep, egrep, fgrep – Searches a file for a pattern

SYNOPSIS

grep [-E | -F] [-c | -l | -q] [-b] [-h] [-i] [-n] [-s] [-v] [-x] [-y] -e *pattern_list*...
[-f *pattern_file*]... [*file*...]

grep [-E | -F] [-c | -l | -q] [-b] [-h] [-i] [-n] [-s] [-v] [-x] [-y] [-e *pattern_list*]...
-f *pattern_file*... [*file*...]

grep [-E | -F] [-c | -l | -q] [-b] [-h] [-i] [-n] [-s] [-v] [-x] [-y] *pattern_list* [*file*...]

Obsolescent version; may not be supported in future releases:

egrep [-c | -l | -q] [-b] [-h] [-i] [-n] [-s] [-v] [-x] [-y] -e *pattern_list*...
[-f *pattern_file*]... [*file*...]

egrep [-c | -l | -q] [-b] [-h] [-i] [-n] [-s] [-v] [-x] [-y] [-e *pattern_list*]...
-f *pattern_file*... [*file*...]

egrep [-c | -l | -q] [-b] [-h] [-i] [-n] [-s] [-v] [-x] [-y] *pattern_list* [*file*...]

fgrep [-c | -l | -q] [-b] [-h] [-i] [-n] [-s] [-v] [-x] [-y] -e *pattern_list*...
[-f *pattern_file*]... [*file*...]

fgrep [-c | -l | -q] [-b] [-h] [-i] [-n] [-s] [-v] [-x] [-y] [-e *pattern_list*]...
-f *pattern_file*... [*file*...]

fgrep [-c | -l | -q] [-b] [-h] [-i] [-n] [-s] [-v] [-x] [-y] *pattern_list* [*file*...]

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4
AT&T extensions (-b, -h, and -y options)

DESCRIPTION

The grep utility searches files for a pattern and prints all lines that contain that pattern. It uses basic regular expressions (BRE), expressions that have string values that use a subset of the possible alphanumeric and special characters, such as those used with ed(1) to match the patterns. A null BRE matches every line.

When the -E option is specified (or the egrep utility is used), grep uses extended regular expressions (ERE) to match the patterns.

`egrep` accepts EREs as in `ed`(1), except for `\(` and `\)`, with the addition of the following:

1. An ERE followed by `+` that matches one or more occurrences of the ERE.
2. An ERE followed by `?` that matches zero or one occurrences of the ERE.
3. An ERE separated by `|` or by a `<newline>` character that match strings that are matched by any of the expressions.
4. An ERE that may be enclosed in parentheses `()` for grouping.

The order of precedence of operators is `[]`, then `* ? +`, then concatenation, then `|` and `<newline>`.

When the `-F` option is specified (or the `fgrep` utility is used), `grep` searches for a string, rather than searching for a pattern that matches an expression.

Because the `$`, `*`, `[`, `^`, `|`, `(`, `)`, and `\` characters in the regular expressions are also meaningful to the shell, it is safest to enclose the entire expression in single quotation marks (`' ... '`).

If files are not specified, `grep` assumes standard input. Usually, each line found is copied to standard output. If more than one input file exists, the file name is printed before each line found.

The `grep` utility accepts the following options:

`-E` Matches using extended regular expressions (ERE). Treat each pattern specified as an ERE. A null ERE matches every line.

`-F` Matches using fixed strings. Treat each pattern specified as a string, rather than a RE. A null string matches every line.

`-b` Precedes each line by the block number on which it was found. Blocks are 512-bytes in size. This can be useful in locating block numbers by context (first block is 0).

`-c` Prints only a count of the lines that contain the pattern.

`-e` *pattern_list*
Specifies one or more patterns to be used during the search for input. Patterns in *pattern_list* are separated by a `<newline>`. To specify a null pattern, use two adjacent `<newline>`s in *pattern_list*. Unless the `-E` or `-F` option is also specified, each pattern is treated as a basic regular expression. If multiple `-e` or `-f` options are specified, all of the specified patterns will be used when matching lines, but the order of evaluation is unspecified.

`-f` *pattern_file*
Reads one or more patterns from the file specified by the path name *pattern_file*. Patterns in *pattern_file* are separated by a `<newline>`. A null pattern can be specified by an empty line in *pattern_file*. Unless the `-E` or `-F` option is also specified, each pattern is treated as a basic regular expression.

`-h` Prevents the name of the file that contains the matching line from being appended to that line. Used when searching multiple files.

`-i`
`-y` Ignores uppercase and lowercase distinction during comparisons.

`-l` Prints the names of files that have matching lines once, separated by `<newline>` characters. Does not repeat the names of files when the pattern is found more than once.

`-n` Precedes each line by its line number in the file (first line is 1).

`-q` Quiet. Nothing is written to the standard output, regardless of matching lines. `grep` exits with zero status if any matches are found, even if an error was detected.

`-s` Suppresses error messages about nonexistent or unreadable files.

`-v` Prints all lines except those that contain the pattern.

`-x` Considers only input lines that use all characters in the line to match an entire fixed string or regular expression to be matching lines.

files The path names of files to be checked.

NOTES

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
`system`, `secadm`	In a privileged administrator shell environment, shell-redirected I/O is not subject to file protections.
`sysadm`	Shell-redirected output is subject to security label restrictions.

If the `PRIV_SU` configuration option is enabled, shell-redirected I/O on behalf of the super user is not subject to file protections.

EXIT STATUS

Exit status is 0 if any matches are found, 1 if none are found, or 2 if an error occurred, even if matches were found (see `-q` option).

EXAMPLES

Example 1: The following command prints on standard output all lines that contain the pattern `Tom Jones` in all files in the current directory:

```
grep 'Tom Jones' *
```

Example 2: The following command prints on standard output all lines within `manuals` file that do not contain the string `User Commands`:

```
grep -v 'User Commands' manuals
```

Example 3: The following command searches for the pattern `User Commands` in the `manuals` file and sends the output to the `users` file:

```
grep 'User Commands' manuals > users
```

SEE ALSO

`ed`(1), `sed`(1), `sh`(1)

`regex`(3C) in the *UNICOS System Libraries Reference Manual*, Cray Research publication SR–2080

NAME

`groups` – Shows group memberships

SYNOPSIS

`/usr/ucb/groups` [*user*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `groups` command shows the groups to which you or the optionally specified user belong. Each user belongs to the groups specified in `/etc/udb`. If you do not own a file but belong to the group by which it is owned, you are granted group access to the file.

When a new file is created, it is given the same group membership as that of the directory in which it is contained.

The `groups` command accepts the following option:

user Name of user that you specify.

NOTES

If this command is installed with the default privilege assignment list (PAL), a user with an active `secadm` category may override mandatory access control (MAC) and discretionary access control (DAC) protections in a privileged administrator shell environment on any file to which input or from which output is being redirected. A user with an active `sysadm` category may override DAC protections in a privileged administrator shell environment on any file to which input or from which output is being redirected, but is constrained by the MAC policy.

Use `id -Gn` to produce the same results as invoking `groups`.

The `groups` command will not be supported after the UNICOS 9.0 release.

EXAMPLES

The following example lists all groups in which you are a member. User input is shown in bold type.

```
$ groups
resrch dev acct pubs
```

FILES

`/etc/udb`	Password file containing login information
`/etc/group`	File containing group names and group IDs

SEE ALSO

`id`(1), `ls`(1)

`setuid`(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

NAME

guest – Performs UNICOS Guest administrative functions

SYNOPSIS

```
guest [-v]
guest -a [-v] [guest_directory]
guest -s [-v] [-T] [-K] [-w] [-n node_name] [-m memsize] [guest_directory]
guest -x [-v] [-o] [-n node_name] [guest_directory]
guest -r [-v] [-n node_name] [guest_directory]
guest -q [-v] [-o] [-n node_name] [guest_directory]
guest -f|-t [-v] [-n node_name] [guest_directory]
guest -d [-v] [-p path] [-n node_name] [guest_directory]
guest -D [-v] [-p path] [guest_directory]
guest -M [-v] [guest_directory]
guest -U [-v] [guest_directory]
guest -c [-v] [-K] [-T] [-n node_name] [-O]
guest -P [-v] name:percent[,name:percent[,...]]
guest -e routine (Can be specified on all guest invocations)
guest -E (Can be specified on all guest invocations)
```

IMPLEMENTATION

Cray PVP systems (except CRAY J90 series and CRAY EL series)

DESCRIPTION

The guest command provides functions to start, status, stop, dump, and change certain attributes of the operating systems running as guests under UNICOS. When specified with no parameters, it will return a text string indicating the *status* of the system on which it is currently running:

HOST Host system with active guests

host Stand-alone system with no guests active

guest Guest system

Except for status functions, all operations (including many of the uppercase options) require an appropriately authorized user.

Nearly all functions operate on a directory containing one or more of the following files or directories:

`guest.rc` A configuration file for the `guest` command. Required if the current directory does not contain both a UNICOS binary file (`unicos`) and a parameter file (`param`). Some command-line options override default settings in this file. The following directories will be searched (in order) for the `guest.rc` file:

guest_directory
`/usr/guest/$LOGNAME/`
`$HOME/`

For information on possible contents, see the FILES section.

`unicos` A UNICOS kernel binary file. Required only if the `quest.rc` file does not specify a binary file.

`param` A UNICOS param file. Required only if the `guest.rc` file does not specify a parameter file.

`crash` A crash binary associated with the UNICOS binary file. Not required.

`kcompress` A program to decompress the UNICOS binary (if necessary). Not required, but if not present, the start function will attempt to use the compression program in `/etc/kcompress`, which may not be compatible with the guest system binary.

`dump` A directory in which dumps will be placed. The directory will be created if it does not exist and if an alternate location is not specified in the `guest.rc` file.

The following command line options are available with most invocations:

`guest_directory`
A directory containing one or more of the files previously mentioned. It can be specified with most of the `guest` command invocations and defaults to the directory search list previously noted.

Status and debug options:

`-v` Lists a verbose `guest` status or status of the command invocation to `stdout`.

`-e` *routine* Provides extra debug output (not recommended).

`-E` Provides extra debug output (not recommended).

Verifying a guest directory:

`-a` This invocation will verify that the current (or specified) guest directory contains enough information to attempt to start a guest system. The system node name is not checked for uniqueness.

Starting a guest system:

`-s` [`-T`] [`-K`] [`-w`] [`-n` *node_name*] [`-m` *memsize*]

This invocation will start a guest system with information provided in the current (or specified) guest directory.

The `-T` option tells the host kernel to produce additional guest-related trace messages in the host and guest kernel.

The `-K` option informs the host system that a panic in the starting guest system should cause a subsequent panic in the host. (The `-T` and `-K` options require the UDB `guestadm` permission and are only useful for guest feature debugging.)

The `-w` option instructs the host to not allow CPUs to enter the guest system until a `guestctl`(2) (`TL RESUME`) call is made by a user program. This option is for internal debugging and is not generally supported.

The `-n node_name` option can be used to change the name of the kernel being started. Both options can be specified in the `guest.rc` file (`NODE_NAME`). The command will exit with a nonzero status if the system could not be started.

Stopping a guest system:

`-x` [`-o`] [`-n` *node_name*]
`-q` [`-o`] [`-n` *node_name*]

You must be the owner of the system that you are trying to stop. (For more information, see the Changing a guest system subsection.) If you do not release the guest memory, you may reload/restart a guest in the same memory, provided that the requested memory does not exceed the current allocation. The `-n` *node_name* option can be used to specify the guest system to be stopped. If a name is not specified on the command line, nor found in the user's `guest.rc` file, the user will be prompted with the names of systems currently owned by them. If the system status indicates that the guest is still in multiuser mode, the command will exit unless the `-o` (oblige) flag is set. The `-q` invocation is equivalent to executing both the stop (`-x`) and release (`-r`) invocations. The command will exit with a nonzero status if no guest system matches the specified criteria.

Releasing a guest system's memory:

`-r` [`-o`] [`-n` *node_name*]

This invocation will release mainframe memory held by the guest. If the guest's status is not *stopped*, the command will exit unless the `-o` (oblige) flag is set. The command will error exit if no guest memory is assigned to the user.

Freezing/Thawing a guest system:

`-f|-t` [`-n` *node_name*]
: In a guest system the `-f` (freeze) flag will only allow CPUs to enter the kernel (no exchanges will be done to a user process in the specified guest.) The `-t` (thaw) flag will reverse this constraint. The dump option uses the freeze/thaw concept internally.

Dumping a guest system:

`-d` [`-p` *path*] [`-n` *node_name*]
: This invocation will attempt to dump the associated guest system memory (and associated host memory) to a file. The dump output file can either be specified with the `-p` *path* option or with the `guest.rc` dump file option. The user will always be prompted for a dump reason to be included in the dump header. The command will exit with a nonzero status if no guest systems are in memory. Dumping a guest will not cause it to stop. Guest user processes will resume executing following the dump. (For more information, see the Freezing/Thawing a guest system subsection.)

`-D` [`-p` *path*]
: Dump all systems. This allows a guest administrator (or `root`) to dump all active systems. If no guest systems are active, only the host is dumped. The invocation requires guest administrator permissions.

Mounting/Unmounting filesystems:

`-M|-U`
: When a guest system is started (`-s`), a set of specified logical devices (see `guest.rc` file description) can be automatically unmounted. Associated `ldcache` is also released at this time. When the associated guest memory is released (`-r`), that same set of logical devices is checked with `fsck`(8) and then remounted. An attempt is made to restore `ldcache` (if previously allocated). The `-M` and `-U` invocations provide a means to mount and unmount (respectively) the file systems in the user's list, separate from starting a guest.

Changing a guest system:

`-c` [`-T`] [`-K`] [`-n` *node_name*] [`-O`]
: This invocation allows administrators to change control information for a running guest system. The `-T` and `-K` options are used for guest debugging. The `-T` option toggles the global guest-related trace flag. The `-K` option will toggle the *kill host on guest panic* flag. The `-n` *node_name* can be used to specify a guest system on which to act. The `-O` (owner) flag will change the owner of the guest system to the current user.

Changing CPU percentages:

`-P` *name:percent*[,*name:percent*[,...]]

The percentage of system CPU resources allocated to each system is, by default, equal to the percentage of mainframe memory occupied by each system or is based on an administrator defaults scheme. This allocation is important only when CPU resources are over-committed. The `-P` option will allow reallocation of the CPU resources across guest systems. A `name:percent` tuple must be included for each active guest and the host. The sum of the percentages must equate to 100. This option is only available to administrative users. Note that the next time a guest system is stopped or started, with the `guest` command, CPU resources will automatically be reallocated based on occupied memory or administrator defaults.

NOTES

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
`system`, `secadm`	Allowed to usurp guest system from the original owner. Locks or unlocks guest communication structures from the command line. Changes CPU percentages of active guests. Sets the guest kernel tracing or the *kill host on guest panic* flag.
`sysadm`	Allowed to use this command to perform all controlling functions except for those functions that require either an active `system` or `secadm` category.

If the `PRIV_SU` configuration option is enabled, the super user or a user with one of the following permission bits set are allowed to perform the following functions:

Permission	Action
`GUESTADMIN`	Allowed to usurp guest system from the original owner. Changes CPU percentages of active guests. Sets the guest kernel tracing or the *kill host on guest panic* flag.
`GUEST`	Allowed to use this command to perform all controlling functions except for those functions that require that the `GUESTADMIN` permission bit be set.

Read and write access to the `/usr/guest` directory can be confined to users in a particular UNICOS group (for example, `guest`) at the discretion of each site.

EXAMPLES

Example 1: Start a guest system from `/usr/guest/joeuser` directory. The directory contains the following files:

```
guest.rc        unicos  param   crash kcompress
% cd /usr/guest/joeuser
% guest -s
gst-45 guest: Info
   The guest kernel binary appears to be compressed.
   Decompressing it with:
        ./kcompress_70

gst-46 guest: Info
   The guest kernel binary decompression was successfully
   completed.

gst-43 guest: Info
   Changing the guest system name from:
        enterprise
   to:
        warpspeed1
/gst70/usr/src is mounted.
        Attempting to unmount...done.

/gst70/usr is mounted.
        Attempting to unmount...done.

/gst70 is mounted.
        Attempting to unmount...done.

gst-53 guest: Info
   A 8 MW Guest system (warpspeed1) has been successfully
   started for user:
        joeuser

gst-55 guest: Info
   The guest (warpspeed1) system console is:
        tty42
   on the host (enterprise) system's OWS.
```

Example 2: Dump a guest system.

```
% guest -d

Enter a dump comment (up to 80 characters):
process hung in guest
gst-23 guest: Info
   Guest dump files will be located below the following directory:
        ./dumps/04050534

Dump segment address:    030412000   cumulative words:    012110000
gst-25 guest: Info
   Successfully completed a guest dump of the system named: warpspeed1

Dump segment address:    077610700   cumulative words:    022703000
gst-25 guest: Info
   Successfully completed a guest dump of the system named: warpspeed1
```

Example 3: Stop a guest system and release its memory.

```
% guest -q
gst-15 guest: Info
   The guest options file:
        guest.rc
   was found in the following directory:
        /usr/guest/joeuser/

gst-16 guest: Info
   Changing the current working directory to:
        /usr/guest/joeuser/

gst-61 guest: Info
   The guest system (warpspeed1) has been successfully stopped.


gst-317 guest: Warning
   There is outstanding I/O on:
        ios: 0
        iop: 0
        chan: 034
        unit: 0
   Waiting for I/O completion.

sys-16 guest: Info
   Device busy
```

```
gst-37 guest: Info
   The guest memory (8 MW) has been returned to host.

/etc/mfsck: Starting pass 1

/etc/mfsck: Starting pass 2

/usr_j: file system opened
/usr_j: super block fname usr_j, fpack sn228
/src_j: file system opened
/src_j: super block fname src_j, fpack sn228
/root_j: file system opened
/root_j: super block fname root_j, fpack sn228
/usr_j: Phase 1 - Check Blocks and Sizes
/root_j: Phase 1 - Check Blocks and Sizes
/src_j: Phase 1 - Check Blocks and Sizes
/root_j: Phase 2 - Visit Directories
/usr_j: Phase 2 - Visit Directories
/root_j: Phase 3 - Checking Directories
/root_j: Phase 4 - Checking Non-Directories and Link Counts
/root_j: Phase 5 - Verify Dynamic Information - (Ignored)
/root_j: Phase 6 - Rebuilding Dynamic Information
/root_j: file system summary
/root_j:        32768 total i-nodes (29073 free i-nodes)
/root_j:        137088 total blocks  (65395 free blocks)
/root_j:  ***** FILE SYSTEM WAS MODIFIED *****
/usr_j: Phase 3 - Checking Directories
/usr_j: Phase 4 - Checking Non-Directories and Link Counts
/usr_j: Phase 5 - Verify Dynamic Information - (Ignored)
/usr_j: Phase 6 - Rebuilding Dynamic Information
/usr_j: file system summary
/usr_j:         32768 total i-nodes (25704 free i-nodes)
/usr_j:         137088 total blocks  (39475 free blocks)
/usr_j:  ***** FILE SYSTEM WAS MODIFIED *****
/src_j: Phase 2 - Visit Directories
/src_j: Phase 3 - Checking Directories
/src_j: Phase 4 - Checking Non-Directories and Link Counts
/src_j: Phase 5 - Verify Dynamic Information - (Ignored)
/src_j: Phase 6 - Rebuilding Dynamic Information
/src_j: file system summary
/src_j:         85712 total i-nodes (45087 free i-nodes)
/src_j:         365040 total blocks  (102446 free blocks)
/src_j:  ***** FILE SYSTEM WAS MODIFIED *****
/etc/mfsck: complete (9 secs.)
```

```
/gst70 is not mounted.
        Attempting to mount /dev/dsk/root_j on /gst70...done.

/gst70/usr is not mounted.
        Attempting to mount /dev/dsk/usr_j on /gst70/usr...done.

/gst70/usr/src is not mounted.
        Attempting to mount /dev/dsk/src_j on /gst70/usr/src...done.
```

Example 4: Example shell script (determines system types).

```
#
# Insure that we are running on the host
# before allowing the application to continue.
#
GUEST=`/etc/guest`
if [ "${GUEST}" = "guest" ]
then
    echo "Running as a guest; do not start"
    echo "production applications"
else
    # your code here
fi
```

FILES

`/usr/guest/Defaults`
: An optional file containing administrator-specified default values for guest systems. Updated by the installation tool.

`/guest`
: A directory containing `crash` and `unicos` binaries for active guests. Do not remove or move binaries placed here by the `guest` command.

`/guest/.diskinfo`
: A directory containing information about file systems unmounted by the `guest` command. Do not remove, move, or change files placed here by the `guest` command.

`/usr/guest/Users`
: An optional file listing individual users and their administrator-specific constraints. Updated by the installation tool.

`guest.rc`
: An optional file containing the default actions and/or the paths to required files such as `unicos` and `param`. Options include the following:

`CRASH`	Location of crash binary
`DUMP_DIRECTORY`	Directory below which system dumps are written (`./dump`)

KCOMPRESS	Location of kernel (de)compression binary (/etc/kcompress)
KERNEL	Location of kernel binary (./unicos)
LOGICAL_DEVICES	List of logical devices to unmount before startup (no default)
MEMSIZE	Requested memory size (value of MAX_GUEST_MEMORY from /usr/guest/Users)
MIN_MEMSIZE	Minimum memory size that you will accept (value of MIN_GUEST_MEMORY from /usr/guest/Default)
NODE_NAME	Node name (system name from the kernel binary)
PARAM	Location of system parameter file (./param)
PARAM_CHECKER	Location of system parameter file checker (/etc/econfig (E)) or (/etc/bconfig (D)) PARAM_CHECKER Location of system parameter file checker (/etc/econfig (E)) or (/etc/bconfig (D))
TTY_CONNECTIONS	Number of tty connections requested (1)

(For more information on your validation, see the /usr/guest/Defaults and /usr/guest/Users files.)

SEE ALSO

udbsee(1)

crash(8), sar(8), shutdown(8) in the *UNICOS Administrator Commands Reference Manual*, Cray Research publication SR–2022

NAME

hash – Remembers or reports utility locations

SYNOPSIS

hash [*utility*...]

hash -r

IMPLEMENTATION

All Cray Research systems

STANDARDS

XPG4

DESCRIPTION

The hash utility affects the way the current shell environment remembers the locations of utilities found. Depending on the arguments specified, it adds utility locations to its list of remembered locations or it purges the contents of the list. When no arguments are specified, it reports on the contents of the list.

Utilities provided as built-ins to the shell are not reported by hash.

The hash utility accepts the following options:

-r Forgets all previously remembered utility locations.

The hash utility accepts the following operands:

utility The name of a utility to be searched for and added to the list of remembered locations. If *utility* contains one or more slashes, the results are unspecified.

The following environment variables affect the execution of hash:

LANG Provides a default value for the internationalization variables that are unset or null. If LANG is unset or null, the corresponding value from the implementation-specific default locale will be used. If any of the internationalization variables contains an invalid setting, the utility will behave as if none of the variables had been defined.

LC_ALL If set to a nonempty string value, override the values of all the other internationalization variables.

LC_CTYPE Determines the locale for the interpretation of sequences of bytes of text data as characters (for example, single- as opposed to multibyte characters in arguments).

LC_MESSAGES Determines the locale that should be used to affect the format and contents of diagnostic messages written to standard error.

NLSPATH Determines the location of message catalogs for the processing of LC_MESSAGES.

PATH Determines the location of *utility*, as described in the XBD specification.

The standard output of hash is used when no arguments are specified. Its format is unspecified, but includes the path name of each utility in the list of remembered locations for the current shell environment. This list consists of those utilities named in previous hash invocations, and may contain those invoked and found through the normal command search process.

Since hash affects the current shell execution environment, it is always provided as a shell regular built-in. If it is called in a separate utility execution environment, such as one of the following:

```
nohup hash -r
find . type f | xargs hash
```

it will not affect the command search process of the caller's environment.

The hash utility may be implemented as an alias, for example, alias -t -, in which case utilities found through normal command search will not be listed by the hash utility.

The effects of hash -r can also be achieved portably by resetting the value of PATH; in the simplest form, this can be:

```
PATH="$PATH"
```

The use of hash with *utility* names is unnecessary for most applications, but may provide a performance improvement of a few implementations; normally, the hashing process is included by default.

EXIT STATUS

The following exit values are returned:

0 Successful completion.

>0 An error occurred.

SEE ALSO

sh(1)

NAME

head – Displays the first few lines of a file

SYNOPSIS

head [-n *number*] [*files*]

Obsolescent version; may not be supported in future releases:

head [-*number*] [*files*]

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The head utility copies its input files to the standard output, ending the output for each file at a designated point. Copying ends at the point in each input file indicated by the -n *number* option (or the -*number* argument). The argument *number* is counted in units of lines.

The head utility accepts the following options and operands:

-*number*
-n *number* The first *number* lines of each input file are copied to the standard output.

files A path name of a file be displayed. If no *file* operands are specified, the standard input is used.

If no options are specified, -n defaults to 10.

If no *files* are specified, head copies lines from the standard input.

When more than one file is specified, the start of each file looks like the following:

==>*filename*<==

EXIT STATUS

The head utility exits with one of the following values:

0 Successful completion.

>0 An error occurred.

SEE ALSO

cat(1), more(1), pg(1), tail(1)

NAME

help – Provides explanation of SCCS messages and commands

SYNOPSIS

help [*args*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The help command is part of the Source Code Control System (SCCS) command set. It finds information to explain a message from a command or explain the use of a command. Zero or more arguments may be supplied.

The help command accepts the following operand:

args The arguments may be either message numbers (which usually appear in parentheses following messages) or command names, of one of the following types:

- Type 1 Begins with nonnumerics, ends in numerics. The nonnumeric prefix is usually an abbreviation for the program or set of routines that produced the message (for example, ge4, for message 4 from the get(1) command).
- Type 2 Does not contain numerics (as a command, such as get(1)).
- Type 3 Is all numeric (for example, 26).

The response of the program will be the explanatory information related to the argument, if there is any.

If help cannot find anything relevant, but the current argument matches the extended regular expression [A-Za-z]+-[0-9]+, it uses the exec command to execute /usr/bin/explain, assuming that the argument is a Cray Research message identifier. Otherwise, it executes man(1) with the argument.

help works across the arguments from left to right, until it finds one that does not appear to be associated with SCCS. At that time, it executes either explain or man, which effectively makes the current argument the last argument processed.

When all else fails, try help stuck.

EXAMPLES

Additional information is obtained from help for messages from SCCS commands as depicted in the following example:

```
$ get -e s.example.c
ERROR [s.example.c]: writable `example.c' exists (ge4)
$ help ge4

ge4:
"writable `...' exists"
For safety's sake, SCCS won't overwrite an existing g-file if it's writable.
If you don't need the g-file, remove it and rerun the get command.
$
```

FILES

`/usr/lib/help`	Directory containing files of message text

SEE ALSO

`admin`(1), `cdc`(1), `comb`(1), `delta`(1), `get`(1), `man`(1), `prs`(1), `rmdel`(1), `sact`(1), `sccsdiff`(1), `unget`(1), `val`(1), `vc`(1), `what`(1)

`sccsfile`(5) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

NAME

host – Looks up host names by using domain server

SYNOPSIS

/usr/ucb/host [-a] [-c *class*] [-d] [-l] [-r] [-t *querytype*] [-v] [-w] *host* [*server*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The host utility looks for information about Internet hosts. It gets this information from a set of interconnected servers that are spread across the country. By default, it maps between host names and Internet addresses. However, with the -t or -a option, it can find all of the information about the specified host that is maintained by the domain server.

The host utility accepts the following options:

-a Operates equivalent to -v -t any (for all).

-c *class* Allows you to specify an address class for the request. Currently, supported types are in, hs, any, and specific numeric values. The default class is in, which represents the Internet address class.

-d Turns on debugging. Network transactions are shown in detail.

-l Produces a listing of a complete domain.

-r Specifies that recursion in the request will be turned off. This means that the name server returns only data it has in its own database; it does not ask other servers for more information.

-t *querytype*
Allows you to specify a particular type of information to be looked up. The arguments are defined in the man page for named(8). Currently supported types are a, ns, md, mf, cname, soa, mb, mg, mr, null, wks, ptr, hinfo, minfo, mx, uinfo, uid, gid, unspec, and the wildcard, which may be written as either any or *. You must specify types in lowercase. The default is to look first for a, and then mx, except that if the verbose option is turned on, the default is only a.

-v Specifies that the printout will be in a verbose format. This is the official domain master file format, which is documented in the man page for named(8). Without this option, output still follows this format in general terms, but an attempt is made to make it more intelligible to typical users. Without -v, a, mx, and cname, records are written as has address, mail is handled by, and is a nickname for, respectively, and time-to-live (TTL) and class fields are not shown.

-w Directs the host to wait for a response. It usually times out after approximately 1 minute.

host Specifies the name or number of the host to be looked up. The program first tries to interpret *host* as a host number. If this fails, it treats it as a host name.

server Specifies a particular server to query. If you do not specify this argument, the default server (usually the local machine) is used.

Specifying Hosts

If you specify *host* as a number, an *inverse query* is done; that is, the domain system looks in a separate set of databases that are used to convert numbers to names. A host number consists of four decimal numbers separated by dots (for example, `128.6.4.194`).

A host name consists of names separated by dots (for example, `topaz.rutgers.edu`). Unless the name ends in a dot, the local domain is appended automatically. Thus, you can specify `host topaz`, and host looks up the address of the machine named `topaz`. If this fails, the name (in this case, `topaz`) is tried unchanged. This same convention is used for mail and other network utilities. They obtain the actual suffix to append by looking at the results of a `hostname` call, and using everything, starting at the first dot. Following is a description of how to customize the host name lookup.

If you specify a name rather than a host number, you can see three different types of output. The following is an example that shows all of them:

```
$ host sun4
sun4.rutgers.edu is a nickname for ATHOS.RUTGERS.EDU
ATHOS.RUTGERS.EDU has address 128.6.5.46
ATHOS.RUTGERS.EDU has address 128.6.4.4
ATHOS.RUTGERS.EDU mail is handled by ARAMIS.RUTGERS.EDU
```

In this example, the user typed the command `host sun4`. The first line indicates that the name `sun4.rutgers.edu` is actually a nickname. The official host name is `ATHOS.RUTGERS.EDU`. The next two lines show the address. (If a system has more than one network interface, there is a separate address for each.)

The last line indicates that `ATHOS.RUTGERS.EDU` does not receive its own mail; mail for it is taken by `ARAMIS.RUTGERS.EDU`. More than one of these lines can exist because some systems have more than one other system that handles mail for them. Technically, each system that can receive mail must have an entry of this type. If the system receives its own mail, there must be an entry that mentions the system itself (for example, "*XXX* `mail is handled by` *XXX*"). However, many systems that receive their own mail do not mention that fact. If a system has a `mail is handled by` entry, but no address, it is not actually part of the Internet, but a system that is on the network will forward mail to it. Systems on Usenet, Bitnet, and several other networks have this type of entry.

BUGS

Unexpected results can be obtained when you type a name; for example, a specification for a host that is not part of the local domain.

The `-l` option tries only the first name server that is listed for the domain that you have requested. If this server is not operational, you may have to specify a server manually. For example, to get a listing of `foo.edu`, you can try `host -t ns foo.edu` to get a list of all the name servers for `foo.edu`, and then try `host -l foo.edu` *xxx* for all *xxx* on the list of name servers until you find one that works.

EXAMPLES

Example 1: The following example gives a listing of all hosts in the `rutgers.edu` domain. Without the `-t` option, `host -l` fetches only address information, which also includes PTR and NS records.

```
host -l rutgers.edu
```

Example 2: The following example gives a complete listing of the zone data for `rutgers.edu`, in the official master file format. (However, the SOA record is listed twice, for arcane reasons.) `-l` is implemented by doing a complete zone transfer and then filtering the information for which you have asked through the `-t` option. The `-t any` option in the following example specifies that all information in the domain is listed; this command must be used only if it is absolutely necessary.

```
host -l -v -t any rutgers.edu
```

SEE ALSO

`nslookup`(1)

`named`(8) in the *UNICOS Administrator Commands Reference Manual*, Cray Research publication SR–2022

NAME

`hostid` – Sets or prints identifier of current host system

SYNOPSIS

`hostid` [*identifier*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `hostid` utility sets or prints the *identifier* of the current host in hexadecimal.

By default, the value of *identifier* is 0. This numeric value is expected to be unique across all hosts and is usually set to the host's Internet address. An appropriately authorized user can set the *identifier* by giving a hexadecimal argument or an address in Internet dot format.

NOTES

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
`system`, `secadm`	Allowed to set the current host ID.
`sysadm`	Allowed to set the current host ID. Shell-redirected I/O is subject to security label restrictions.

If the `PRIV_SU` configuration option is enabled, the super user is allowed to set the current host ID.

SEE ALSO

`gethostid`(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

NAME

hostname – Prints the name of current host system

SYNOPSIS

hostname [*nameofhost*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The hostname utility prints the name of the current host system. An appropriately authorized user can set the host name by giving a *nameofhost* argument to hostname.

NOTES

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
system, secadm	Allowed to set the current host name.
sysadm	Allowed to set the current host name. Shell-redirected I/O is subject to security label restrictions.

If the PRIV_SU configuration option is enabled, the super user is allowed to set the current host name.

SEE ALSO

gethostname(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

UNICOS Networking Facilities Administrator's Guide, Cray Research publication SG–2304

NAME

hpm – Monitors hardware performance during program execution

SYNOPSIS

hpm [-d] [-g *group*] [-o *file*] [-p] [-r] [-V] *program* [*args*]

IMPLEMENTATION

Cray PVP systems

DESCRIPTION

The hpm utility monitors machine performance while your program executes. Your program can be written in any language available under the UNICOS operating system, though it cannot make use of Parallel Virtual Machine (PVM) message-passing software. hpm gives results for only whole programs and writes its output to standard error.

The hpm utility accepts the following options:

-d Displays additional rates as though run in dedicated mode; some rates are based on wall-clock time, not CPU time.

These times give a rough estimate of concurrency when the user's program is autotasked, microtasked, or macrotasked. The displayed rates should not be considered to be exact. The extra wall-clock time required to execute the code to be monitored, combined with the extra wall-clock time needed to notify the hpm utility that the user's command is done, can easily exceed the wall-clock time needed to actually execute the code.

If you need more precise measurements of execution time versus wall-clock time, use the atexpert(1) command or the ja(1) command. It is possible to run your program on a dedicated machine or under the ded(8) command. However, even running under ded or on a dedicated machine still cannot guarantee exact dedicated timings.

-g *group* Number of the hardware monitor group to be used. This option does not apply to the CRAY C90 or the CRAY T90 series because they have only one Hardware Performance Monitor (HPM) group. The *group* argument can be one of the following values:

- 0 Execution summary (default)
- 1 Hold issue conditions
- 2 Memory activity
- 3 Vector events and instruction summary

-o *file* By default, hpm writes its output to standard error. If this option is specified, hpm writes its output to the file name specified as *file*.

-p Displays results for just the program and excludes all performance information for any child processes.

`-r` Generates a raw-format output, suitable for postprocessing by tools such as `perfview`(1) and `awk`(1). See the *Guide to Parallel Vector Applications*, Cray Research publication SG–2182, for a description of this format.

`-V` Displays the current version of `hpm`, as well as a short copyright notice.

program Executable file to be run.

args Arguments to *program*.

NOTES

HPM does not work with Parallel Virtual Machine (PVM) code on Cray PVP systems.

The default counter hardware monitor group is 0, the group most commonly run by typical users (does not apply to the CRAY C90 or CRAY T90 series).

The performance utility Perftrace also uses the hardware performance monitor device. Using the `libperf.a` library with `hpm` may generate unusable results for both.

On Cray PVP (except CRAY C90 and CRAY T90 series) systems, groups 0 and 3 report megaflop rates. By default, these megaflop rates do not reflect concurrent execution of an autotasked, microtasked, or macrotasked program, since they are calculated per CPU second and not per wall-clock second. When running a multitasked program, you can obtain more informative rates that use wall-clock seconds by specifying the `-d` option. However, note the special considerations for using this option, as described under the `-d` option previously.

The meanings of the HPM statistics and their implications are discussed in detail in the *Guide to Parallel Vector Applications*, Cray Research publication SG–2182.

Because of variations in system overhead and other factors, `hpm` statistics may not be precisely repeatable. The statistics gathered and displayed by the `hpm` utility should not be construed as accounting information, nor should they be considered to be exact.

EXAMPLES

Example 1: For Cray PVP systems, the following examples execute a program four times to receive all four hardware monitor groups in the file `prog.hpm`. The last example shows that only one program execution is needed on the CRAY C90 and CRAY T90 series.

Standard shell or Korn shell (except on the CRAY C90 or CRAY T90 series):

```
$ f90 prog.f
$ hpm -g0 ./a.out 2>> prog.hpm
$ hpm -g1 ./a.out 2>> prog.hpm
$ hpm -g2 ./a.out 2>> prog.hpm
$ hpm -g3 ./a.out 2>> prog.hpm
```

C shell (except on the CRAY C90 or CRAY T90 series):

```
% f90 prog.f
% ( hpm -g0 ./a.out ) > & prog.hpm
% ( hpm -g1 ./a.out ) >> & prog.hpm
% ( hpm -g2 ./a.out ) >> & prog.hpm
% ( hpm -g3 ./a.out ) >> & prog.hpm
```

CRAY C90 or CRAY T90 series (all shells):

```
$ f90 prog.f
$ hpm ./a.out
```

Example 2: The following standard shell example (not for the CRAY C90 or CRAY T90 series) generates raw-format data and then processes the output with the perfview(1) command. After these command lines, perfview(1) runs interactively.

```
$ cc prog.c
$ hpm -g0 -r ./a.out 2> raw.data
$ hpm -g3 -r ./a.out 2>> raw.data
$ perfview raw.data
```

Example 3: The following example (not for the CRAY C90 or CRAY T90 series) writes the hpm data to a file for processing by perfview(1):

```
$ cc prog.c
$ hpm -g0 -o data.0 -r ./a.out
$ hpm -g3 -o data.3 -r ./a.out
$ cat data.0 data.3 >perf.data
$ perfview
```

SEE ALSO

atexpert(1), awk(1), csh(1), ja(1), perfview(1), sh(1)

hpm(4) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

performance(7), perftrace(7) (available only online)

ded(8) in the *UNICOS Administrator Commands Reference Manual*, Cray Research publication SR–2022

Guide to Parallel Vector Applications, Cray Research publication SG–2182, for descriptions of all the performance tuning tools

The following manuals are Cray Research Proprietary; dissemination of this documentation to non-CRI personnel requires approval from the appropriate vice president and a nondisclosure agreement. Export of technical information in this category may require a Letter of Assurance.

NAME

iconv – Codeset conversion

SYNOPSIS

iconv -f *fromcode* -t *tocode* [*file...*]

IMPLEMENTATION

All Cray Research systems

STANDARDS

XPG4

DESCRIPTION

The iconv utility converts the encoding of characters in *file* from one codeset to another and writes the results to standard output.

Character encodings in either codeset may include single-byte values (for example, for the ISO 8859-1:1987 standard characters) or multibyte values (for example, for certain characters in the ISO 6937:1983 standard). The results of specifying invalid characters in the input stream (either those that are not valid members of the *fromcode* or those that have no corresponding value in *tocode*) are specified in the system documentation.

The iconv utility accepts the following options and operands:

-f *fromcode* Identifies the codeset of the input file. Valid values for *fromcode* are specified in the system documentation.

-t *tocode* Identifies the codeset to be used for the output file. Valid values for *tocode* are specified in the system documentation.

file A path name of the input file to be translated. If *file* is omitted, the standard input is used.

The following environment variables affect the execution of iconv:

LANG Provides a default value for the internationalization variables that are unset or null. If LANG is unset or null, the corresponding value from the implementation-specific default locale will be used. If any of the internationalization variables contains an invalid setting, the utility will behave as if none of the variables had been defined.

LC_ALL If set to a nonempty string value, overrides the values of all the other internationalization variables.

LC_CTYPE Determines the locale for the interpretation of sequences of bytes of text data as characters (for example, single- as opposed to multibyte characters in arguments). During translation of the file, this variable is superseded by the use of the *fromcode* option-argument.

LC_MESSAGES Determines the locale that should be used to affect the format and contents of diagnostic messages written to standard error.

NLSPATH Determines the location of message catalogs for the processing of LC_MESSAGES.

EXIT STATUS

The following exit values are returned:

0 Successful completion.

>0 An error occurred.

EXAMPLES

The following example converts the contents of file mail.x400 from the ISO 6937:1983 standard codeset to the ISO 8859-1:1987 standard codeset, and stores the results in the file mail.local:

```
iconv -f IS6937 -t IS8859 mail.x400 > mail.local
```

SEE ALSO

gencat(1)

NAME

`id` – Prints user and group IDs and names

SYNOPSIS

`id` [*user*]
`id -G` [`-n`] [*user*]
`id -g` [`-n`] [`-r`] [*user*]
`id -u` [`-n`] [`-r`] [*user*]
`id -a` [*user*]

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4
CRI extensions (`-a` option)

DESCRIPTION

If no *user* is specified, the `id` utility writes the user and group IDs and the corresponding user and group names of the invoking process to standard output. When the effective and real IDs do not match, both are written.

If *user* is specified and the invoking process has the appropriate privileges, the user and group IDs of the selected user are written. In this case, effective IDs are assumed to be identical to real IDs.

If multiple groups are supported by the underlying system, the supplementary group affiliations are also written.

The `id` utility accepts the following options:

`-a` Prints the current account ID and name.

`-g` Prints only the effective group ID.

`-G` Prints all different group IDs (effective, real, and supplementary) only. If more than one distinct group affiliation exists, the utility prints each affiliation.

`-n` Prints the name of the user or group, rather than the numeric ID.

`-r` Prints the real ID, rather than the effective ID.

`-u` Prints the only the effective user ID.

NOTES

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
`system`, `secadm`	In a privileged administrator shell environment, allowed to write shell-redirected output to any file.
`sysadm`	Shell-redirected output is subject to security label restrictions.

If the `PRIV_SU` configuration option is enabled, the super user can write shell-redirected output to any file.

WARNINGS

The `id -Gn` command produces the same output as `/usr/ucb/groups`. Users should use the `id` utility because the `groups` utility may be removed in a future UNICOS release.

EXIT STATUS

The `id` utility exits with one of the following values:

0 Successful completion.

>0 An error occurred.

EXAMPLES

The following example prints your user and group IDs and names:

```
$ id
uid=1054(mary) gid=101(acct) groups=508(allgrp),717(compilers),24(source)
```

FILES

`/etc/udb`	User validation file that contains user control limits
`/etc/group`	Group files that contain group names and group IDs

SEE ALSO

`groups`(1B)

`getuid`(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

`group`(5), `udb`(5) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

NAME

ipcrm – Removes a message queue, semaphore set, or shared memory ID

SYNOPSIS

ipcrm [-m *shmid*] [-M *shmkey*] [-q *msqid*] [-Q *msgkey*] [-s *semid*] [-S *semkey*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The ipcrm command removes one or more message queues, semaphore sets, or shared memory identifiers.

The ipcrm command accepts the following options:

-m *shmid* Removes the shared memory identifier *shmid* from the system. The shared memory segment and data structure associated with it are destroyed after the last detach operation.

-M *shmkey* Removes the shared memory identifier, created with key *shmkey*, from the system. The shared memory segment and data structure associated with it are destroyed after the last detach operation.

-q *msqid* Removes the message queue identifier *msqid* from the system and destroys the message queue and data structure associated with it.

-Q *msgkey* Removes the message queue identifier, created with key *msgkey*, from the system and destroys the message queue and data structure associated with it.

-s *semid* Removes the semaphore identifier *semid* from the system and destroys the set of semaphores and data structure associated with it.

-S *semkey* Removes the semaphore identifier, created with key *semkey*, from the system and destroys the set of semaphores and data structure associated with it.

The details of the remove operations are described in msgctl(2), shmctl(2), and semctl(2). Use the ipcs(1) command to find the identifiers and keys.

NOTES

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
system, secadm	Allowed to remove any identifier.
sysadm	Allowed to remove any identifier, subject to security label restrictions on the identifier's path. Shell-redirected I/O is subject to security label restrictions.

If the PRIV_SU configuration option is enabled, the super user is allowed to remove any identifier.

SEE ALSO

ipcs(1)

msgctl(2), msgget(2), msgrcv(2), msgsnd(2), semctl(2), semget(2), semop(2), shmat(2), shmctl(2), shmdt(2), shmget(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

stdipc(3C) in the *UNICOS System Libraries Reference Manual*, Cray Research publication SR–2080

ipc(5), msg(5), sem(5), shm(5) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

ipc(7) Online only

NAME

ipcs – Reports interprocess communication (IPC) facilities status

SYNOPSIS

ipcs [-a] [-b] [-c] [-e] [-m] [-o] [-p] [-q] [-s] [-t]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The ipcs command prints information about active interprocess communication (IPC) facilities. Without options, information is printed in short format for message queues, shared memory, and semaphore sets that are currently active in the system.

The information that is displayed is controlled by the options supplied.

-m Prints information about active shared memory segments.

-q Prints information about active message queues.

-s Prints information about active semaphore sets.

If -q, -m, or -s are specified, information about only those indicated is printed. If none of these three are specified, information about all three is printed subject to the following options. For detailed information about an ipcs listing, see the ipcs Listing Information section.

-a Uses all print options. (This is a shorthand notation for -b, -c, -o, -p, and -t.)

-b Prints information on biggest allowable size: maximum number of bytes in messages on queue for message queues, size of segments for shared memory, and number of semaphores in each set for semaphores.

-c Prints creator's login name and group name.

-e Provides the access control list (ACL) flag, security level, and compartment flag as the fields immediately following the mode field. When at least one compartment is set, the facility's compartment flag is displayed as a plus sign (+) adjacent to the facility's security level. When a facility has an associated ACL, its ACL flag appears as a letter a adjacent to the mode field. A facility that has a wildcard security level has an asterisk (*) displayed for its security level.

-o Prints information on outstanding usage: number of messages on the queue and total number of bytes in those messages and number of processes attached to shared memory segments.

-p Prints process number information: process ID of last process to send a message, process ID of last process to receive a message on message queues, process ID of creating process, and process ID of last process to attach or detach on shared memory segments.

`-t` Prints time information: time of the last control operation that changed the access permissions for all facilities, time of last `msgsnd`(2) and last `msgrcv`(2) on message queues, time of last `shmat`(2) and last `shmdt`(2) on shared memory, time of last `semop`(2) on semaphores.

`ipcs` Listing Information

This section lists the column headings in an `ipcs` listing and describes the information produced by the `ipcs` command. The default headings and informtion produced by this command are as follows except for those described with options in parentheses (for example, `CREATOR`). In these exceptions, the options named cause the corresponding heading to appear.

Heading	Description
`T`	Type of the facility: `q` Message queue `m` Shared memory segment `s` Semaphore
`ID`	The identifier for the facility entry.
`KEY`	The key used as an argument to `msgget`(2), `semget`(2), or `shmget`(2) to create the facility entry. NOTE: The key of a shared memory segment is changed to `IPC_PRIVATE` when the segment has been removed until all processes attached to the segment detach it.
`MODE`	The facility access modes and flags: The mode consists of 12 characters that are interpreted as follows. The first three characters are one of the following: `P` The persistent facility is enabled for the message queue, semaphore set, or shared memory segment. `R` A process is waiting on a `msgrcv`(2) operation. `S` A process is waiting on a `msgsnd`(2) operation. `-` The corresponding special flag is not set. The next 9 characters are interpreted as three sets of 3 bits each. The first set refers to the owner's permissions; the next set refers to permissions of others in the user group of the facility entry; and the last set refers to all others. Within each set, the first character indicates permission to read, the second character indicates permission to write or alter the facility entry, and the last character is currently unused. The permissions are indicated as follows: `r` Read permission is granted. `w` Write permission is granted. `a` Alter permission is granted. `-` The indicated permission is not granted.

OWNER The login name of the owner of the facility entry.

GROUP The group name of the group of the owner of the facility entry.

CREATOR (-a, -c) The login name of the creator of the facility entry.

CGROUP (-a, -c) The group name of the group of the creator of the facility entry.

CBYTES (-a, -o) The number of bytes in messages currently outstanding on the associated message queue.

QNUM (-a, -o) The number of messages currently outstanding on the associated message queue.

QBYTES (-a, -b) The maximum number of bytes allowed in messages outstanding on the associated message queue.

LSPID (-a, -p) The process ID of the last process to send a message to the associated queue.

LRPID (-a, -p) The process ID of the last process to receive a message from the associated queue.

STIME (-a, -t) The time the last message was sent to the associated queue.

RTIME (-a, -t) The time the last message was received from the associated queue.

CTIME (-a, -t) The time when the associated entry was created or changed.

NATTCH (-a, -o) The number of processes attached to the associated shared memory segment.

SEGSZ (-a, -b) The size of the associated shared memory segment.

CPID (-a, -p) The process ID of the creator of the shared memory entry.

LPID (-a, -p) The process ID of the last process to attach or detach the shared memory segment.

ATIME (-a, -t) The time the last attach was completed to the associated shared memory segment.

DTIME (-a, -t) The time the last detach was completed on the associated shared memory segment.

NSEMS (-a, -b) The number of semaphores in the set associated with the semaphore entry.

OTIME (-a, -t) The time the last semaphore operation was completed on the set associated with the semaphore entry.

NOTES

Things can change while `ipcs` is running; the information it gives is guaranteed to be accurate only when it was retrieved.

Only an appropriately authorized user can see output for IPC facilities whose active security label is greater than that of the user.

If this command is installed with a privilege assignment list (PAL), a user who is assigned the following privilege text upon execution of this command is allowed to perform the action shown:

Privilege Text	Action
`showall`	Allowed to see output for all IPC facilities.

If the PRIV_SU configuration option is enabled, the super user is allowed to see output for all IPC facilities.

FILES

/dev/Rmem	Kernel data structures
/etc/group	Group names
/etc/passwd	User names
/etc/udb	User database (UDB) information
/etc/udb.public	User database (UDB) public information

SEE ALSO

ipcrm(1)

msgctl(2), msgget(2), msgrcv(2), msgsnd(2), semctl(2), semget(2), semop(2), shmat(2), shmctl(2), shmdt(2), shmget(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

stdipc(3C) in the *UNICOS System Libraries Reference Manual*, Cray Research publication SR–2080

ipc(5), msg(5), sem(5), shm(5) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

ipc(7) Online only

NAME

`ispell` – Corrects spelling for a file

SYNOPSIS

`ispell` [*file...*]
`ispell` [`-l` | `-D` | `-E`]
`spell` [`+local_file`] [*file...*]

IMPLEMENTATION

All Cray Research systems

STANDARDS

FSF

DESCRIPTION

The `ispell` program helps you to correct typos in a file, and to find the correct spelling of words. When presented with a word that is not in the dictionary, `ispell` offers possibilities.

The best way to use `ispell` is with GNU Emacs. For documentation about this mode, see the info topic "`ispell`."

`ispell` can also be used by itself. In this case, the most common usage is "`ispell` *filename*." If `ispell` finds a word that is not in the dictionary, the word is printed at the top of the screen. `ispell` then checks the dictionary for near misses (words that differ only by a single letter, a missing or extra letter, or a pair of transposed letters). Any that are found are printed on the following lines, and two lines of context containing the word are printed at the bottom of the screen. If your terminal can display reverse video, the word is highlighted.

If you think the word is correct as is, you can press the `<space>` key to accept it this one time, the `<a>` key to accept it for the rest of this file, or the `<i>` key to accept it and put it in your private dictionary. If one of the near misses is the word you want, type the corresponding number. You can press the `<r>` key and you will be prompted for a replacement word. The string you type will be broken into words, and each one will also be checked. You can also press the `<?>` key for help.

`ispell` accepts the following options:

`-l` Produces a list of misspelled words from the standard output. This mode is compatible with the traditional spell program, except that the output is not sorted.

`-D` Prints words with flags.

`-E` Prints expanded words as follows:

```
% ispell
word: independant
how about: independent
word: ^D
```

`-u` `ispell` tries to be compatible with the traditional spell program.

file Name of file to be corrected.

If `ispell` is started with no arguments, it enters a loop reading words from the standard input and printing messages about them on the standard output. You can use this mode to find the spelling of a problem word.

There are several other options provided so that other programs can use `ispell`. See the documentation in the `ispell` source directory for details.

If `ispell` is executed by using the name `spell`, it tries to be compatible with the traditional spell program. You can also get this behavior by specifying the `-u` option. In this case, the list of files (or standard input) is checked, and an alphabetized list of misspellings is produced on the standard output.

FILES

`/usr/lib/ispell/ispell.dict`	System dictionary
`$HOME/ispell.words`	Private dictionary

SEE ALSO

`emacs`(1)

`/usr/lib/emacs/info/ispell.texinfo`
/

NAME

ja – Job accounting information

SYNOPSIS

ja [[-f] [-o] [-s [-e]] [[-a *acid*] [-d] [-D] [-g *d*] [-j *jid*] [-l [-C] [-h]] [-n *names*] [-p *marks*] [-r] [-u *uid*]]] [-t] [*file*]

ja [-m] [*file*]

ja [[-c[-h]] [-f] [-s [-e]] [[-a *acid*] [-d] [-D] [-g *gid*] [-j *jid*] [-l[-C-h]] [-n *names*] [-p *marks*] [-r] [-u *uid*]]] [-t] [*file*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The ja command provides job- or session-related accounting information. This information is taken from the job accounting file to which the kernel writes, provided that job accounting is enabled. The job accounting file can be either the *file* you provide or the default, described in the following. ja provides information only about terminated processes. The login shell and the current ja command being executed are active processes and are not reported by ja. See ps(1) for information about active processes.

To enable job accounting, use the ja command. You may specify only the mark option (-m) and the optional *file* name when enabling. If the job accounting file does not exist, ja creates it. If the file does exist, accounting information is appended to the existing file. If job accounting is already enabled and the optional file name specified is a file other than the currently active job accounting file, the newly specified file becomes the job accounting file.

If you do not specify the optional file name, a default name of the following form is used:

$TMPDIR/.jacct*job ID*

The TMPDIR environment variable is not exported in at(1) or crontab(1) jobs. You must specify the job accounting file name in the at(1) or crontab(1) commands; otherwise, ja will abort.

On normal termination of job accounting (-t specified), ja removes the job accounting file and disables job accounting. If you specify the optional file name when enabling, specify the same name when terminating.

The ja command lets you mark the positions of various commands (processes) by writing the position of the next accounting record to be processed to standard output. You can use these marks when generating reports to restrict the information reported.

There are three groups of options you can use with the `ja` command:

Report selection options
([[`-c`], [`-f`], [`-o`], and [`-s`])
Mark and disable options
([`-m`] or [`-t`])
Report modifier options
([`-d`], [`-e`], [`-h`], [`-l`], [`-r`], [[`-a` *acid*], [`-g` *gid*], [`-j` *jid*], [`-n` *names*], [`-p` *marks*], [`-u` *uid*]]], [`-C`], and [`-D`])

Report Selection Options

The `ja` command can produce four kinds of reports by using the `-c`, `-f`, `-o`, and `-s` options; these are first summarized and then described in detail.

In summary, the four report selection options are as follows:

`-c` Produces the command report (`-c` and `-o` are mutually exclusive).

`-f` Produces the command flow report.

`-o` Produces the other (alternative) command report (`-o` and `-c` are mutually exclusive).

`-s` Produces the summary report.

Described in detail, the four report selection options are as follows:

`-c` Produces a command report. The following fields are reported when you specify the `-c` option with the `-l` option or with the `-l` and `-h` options. These fields provide statistics about individual processes.

`Command Name`	First 8 characters of the name of the command that was executed.
`Started At`	Start time of the process.
`Elapsed Seconds`	Elapsed time of the process.
`User CPU Seconds`	Amount of CPU time the process consumed while it was executing in user mode.
`Sys CPU Seconds`	Amount of CPU time the process consumed while it was executing in system mode.
`I/O Wait Sec Lck`	Amount of time the process waits for I/O while it is locked in memory. I/O wait time is the time a process is blocked until it is rescheduled. The process is blocked while waiting for things such as raw I/O to complete.
`I/O Wait sec Unlck`	Amount of time the process is blocked until it is rescheduled while it is not locked in memory. Time spent for system buffers and buffered I/O blocks are included.

`CPU Mem Avg Mwds` Average amount of memory that this process used. This value is calculated by dividing the memory integral by the total CPU time (*system + user CPU time*). For more information about memory integrals, see *UNICOS Resource Administration*, Cray Research publication SG–2302.

`I/O WMem Avg Mwds` Average amount of memory that the process used while it was locked in memory and waiting for I/O. The value is calculated by dividing the I/O wait memory integral by the I/O wait time while locked in memory.

`Kwords Xferred` Number of characters read or written by the `read`, `write`, `reada`, `writea`, and `listio` system calls (see `read`(2), `write`(2), `reada`(2), `writea`(2), and `listio`(2)).

`Log I/O Request` Number of logical I/O requests that the process performed. A logical I/O request is performed each time a process calls a `read`, `write`, `reada`, or `writea` system call. When the `listio` system call (see `listio`(2)) is called, the number of logical I/O requests is equal to the number of strides multiplied by the number of requests processed.

`Phy I/O Request` Number of times data actually was read from or written to a device. This count does not include requests found in the buffer cache and requests retrieved along with another I/O request.

`Memory HiWater` Maximum amount of memory the process used at any one time. The value is reported in units of 512 words.

`Ex St` Lower 8 bits from the exit status of the process. See `wait`(2) for more information.

`Ni` The last nice value of the process; reported when the total CPU time (*user + system CPU time*) is less than 1 second. If the total CPU time is greater than or equal to 1 second, the minimum nice value at 1 second and onward is listed. The minimum nice value corresponds to the highest priority the process was niced.

`Fl` Accounting flag. The following values are available:

`F` The process forked but did not execute.

`S` The process used super-user privileges.

The accounting flags are defined in `/usr/include/sys/acct.h`.

`SBUs` System billing unit (SBU) for the process. The system administrator configures SBU calculations. For more information, see *UNICOS Resource Administration*, Cray Research publication SG–2302.

When a process is multitasked, the `-c` option produces reports that contain the following fields. Additional lines of multitasking information follow the per-process statistics.

`Command Name` Number of processors that were connected to the process. For example, a value of `# 1 CPU` indicates that the line contains information about the process when only one CPU was connected to it. Likewise, `# 2 CPU` denotes information when two CPUs were connected.

`User CPU Seconds` User CPU time (in seconds) when the process was connected to the number of CPUs specified in the `Command Name` field. The user CPU time reported in this field includes wait semaphore time.

Example:

```
Command     Started     Elapsed     User CPU
 Name          At       Seconds      Seconds
-------     -------     -------     --------

 a.out     16:19:18     1.1251       5.6487
# 1 CPU                             0.0122
# 2 CPU                             0.0194
# 3 CPU                             0.0125
# 4 CPU                             0.0028
# 5 CPU                             0.0120
# 6 CPU                             0.0108
# 7 CPU                             0.8371
# 8 CPU                             4.7813
```

In this example, `a.out` is a multitasked program. The sum of the multitasked user CPU time is `5.6881`, which is larger than `5.6487`, the user CPU time reported for `a.out`. The larger number reflects the fact that the multitasked user CPU time includes wait semaphore time.

To calculate the wait semaphore time, subtract the per-process user CPU time from the sum of the multitasked user CPU times. In the previous example, the wait semaphore time is 5.6881 – 5.6487 = 0.0394 seconds.

The `-c` and `-d` options produce the following additional fields that contain information about device-specific I/O for the process in the preceding line.

`Command Name` Logical device accounting name. The name may span many fields. An example value is `# Block device dd29`.

`Kwords Xferred` Number of characters read or written by the `read`(2), `write`(2), `reada`(2), `writea`(2), and `listio`(2) system calls to the device specified in the `Command Name` field.

`Log I/O Request` Number of the `read`(2), `write`(2), `reada`(2), and `writea`(2) system calls made to the device. When `listio`(2) is called, the number of logical I/O requests is equal to the number of strides multiplied by the number of requests processed on the device.

`-f` Produces a command flow report. This report provides information on the parent/child relationships of processes and, if you specify the `-l` option, CPU user and system time (in seconds).

`-o` Produces an alternative (other) command report. The `-o` option report contains the following fields, which show statistics about individual processes. Several fields show significant values only if performance accounting has been enabled; otherwise, the string `NA` is printed.

`Command Name`	First 8 characters of the name of the command that was executed.
`Started At`	Start time of the process.
`Elapsed Seconds`	Elapsed time of the process.
`Proc ID`	Process ID of the current process.
`Parent ProcID`	Process ID of the parent process.
`Sys Call Seconds`	System call time (in seconds).
`I/O Wait Secs Term`	I/O terminal wait time; I/O wait time is the period of time starting when a process is blocked and ending when it is rescheduled. This field contains a significant value only if performance accounting is enabled (see `devacct`(8)).
`Wait Swap Seconds`	Time (in seconds) that the process waited while swapped out of memory. This field contains a significant value only if performance accounting is enabled (see `devacct`(8)).
`Number of Swaps`	Number of swaps for the current process.
`Phy Blks Mvd: Buf`	Number of physical blocks transferred by the process from or to a block device by using the system buffer I/O interface. This field contains a significant value only if performance accounting is enabled (see `devacct`(8)).
`Phy Blks Mvd: Raw`	Number of physical blocks transferred by the process to and from a block device by using the raw I/O interface. This field contains a significant value only if performance accounting is enabled (see `devacct`(8)).
`Memory Hiwater`	Maximum amount of memory the process used at any one time. The value is reported in units of 512 words.
`Start fract`	Clocks since last second mark was displayed. This field contains a significant value only if performance accounting is enabled (see `devacct`(8)).

`-s` Produces a summary report. The `-s` option report contains the following fields, which provide accumulated usage statistics for the reporting period.

`Job Accounting File Name`
: Name of the file to which the kernel writes the `ja` accounting records.

`Operating System`
: Operating system name, node name, release, version, and hardware type.

`User Name (ID)`
: Name and user ID of the real user.

`Group Name (ID)`
: Name and group ID of the real group.

`Account Name (ID)`
: Account name and account number that this process uses. Multiple account ID usage is listed, but not individual accounts.

`Job ID`
: Job ID associated with these processes.

`Report Starts`
: Starting time of the process that began first during the reporting period.

`Report Ends`
: Ending time of the process that was the last to complete during the reporting period.

`Elapsed Time`
: Duration of the reporting period in seconds (the difference between the report ending and starting times).

`User CPU Time`
: Total CPU time (in seconds) used during the reporting session while the processes were in user mode. (This field is expanded to report multitasking data.)

`System CPU Time`
: Total CPU time (in seconds) used during the reporting session while the processes were in system mode.

`I/O Wait Time (Locked)`
: Cumulative time (in seconds) the system spent waiting for I/O while the processes were locked in memory.

`I/O Wait Time (Unlocked)`
: Cumulative time (in seconds) the system spent waiting for I/O while the processes were not locked in memory.

`CPU Time Memory Integral (Mword-second)`
: Sum of the memory integrals for all processes. For more information on memory integrals, see *UNICOS Resource Administration*, Cray Research publication SG–2302.

`SDS Time Memory Integral`
: (Not on CRAY EL series systems) Measure of SDS use with respect to how long the SDS space was used.

`I/O Wait Time Memory Integral (Mword-second)`
: Measure of how much memory was used when the processes waited for I/O while locked in memory.

`Data Transferred` Total number of characters read or written by the `read`(2), `write`(2), `reada`(2), `writea`(2), and `listio`(2) system calls by all processes in the reporting period.

`Maximum memory used (Mword)`
Maximum amount of memory used by any process at one time.

`Logical I/O Requests`
Total number of `read`(2), `write`(2), `reada`(2), and `writea`(2) system calls executed by all processes in the reporting period. The sum of the number of strides multipled by the number of requests processed for each `listio`(2) call is added to the logical I/O request total.

`Physical I/O Requests`
Total number of times data was read to or written from a physical device by all processes in the reporting period.

`Number of Commands` Total number of commands that completed during the reporting period.

`Billing Units` Sum of the system billing units (SBUs) of all processes.

If a process uses a massively parallel processor (where there is a Cray MPP system), the report contains the following additional information:

`MPP Time` Total number of CPU seconds that the Cray MPP system was used.

`MPP Barrier Bits` Total number of barrier bits used and the largest number used at one time.

`MPP Processor Elements`
Total number of processing elements (PEs) used and the largest number used at one time.

If a process is multitasked, the `User CPU Time` section of the report using the `-s` option expands to include the following information:

`User CPU Time` Total amount of user CPU time (seconds) used during the reporting session while the processes were in user mode.

If the system administrator has defined weighting factors for multiple CPU use, a weighted user CPU time is also reported. This value is in brackets. It is calculated by taking the sum of the weight multiplied by the user CPU time for all concurrent CPUs used. In this instance, the user CPU time includes the wait semaphore time.

In Example 3 of the EXAMPLES section, the total user CPU time (excluding wait semaphore time) is 3.5826 seconds. The weighted user CPU time is 3.4636 seconds. Therefore, an incentive exists to use multitasking with this program.

`Multitasking Breakdown`
: The first part of the breakdown shows the user CPU usage by the number of concurrent CPUs used. The `Weight` and `Weighted seconds` columns appear only if the system administrator has defined CPU weighting factors.

`Concurrent CPUs`
: Number of processors simultaneously connected to the process(es). A connected processor may be executing real work, or it may be waiting on a blocked condition, such as a semaphore.

`Weight`
: The weighting factor for having *n* CPUs connected to the process(es), where *n* is given by the previous field. To determine the factor for the *n*th CPU, subtract the current weighting factor from the one in the previous line, if there is one. In Example 3 of the EXAMPLES section, the second CPU is 90% as expensive as the first CPU.

`Connect Seconds`
: The number of seconds that N CPUs were connected to the processes.

`CPU Seconds`
: The number of user CPU seconds used by *n* concurrent processors. It includes wait semaphore time and is the product of the number of concurrent CPUs multiplied by the connect seconds.

`Weighed Seconds`
: Weighted CPU seconds; the product of the weight and the CPU seconds.

The final portion of the breakdown shows multitasking summary statistics.

`Concurrent CPUs (Avg.)`
: Average number of concurrent CPUs that were connected to the process(es). It is calculated by dividing the total CPU seconds by the total connect seconds.

`Weight (Avg.)`
: Average weighting factor. It is calculated by dividing the total weighted seconds by the total connect seconds.

`Connect seconds (total)`
: Sum of the connect seconds found in the first portion of the breakdown.

`CPU seconds (total)`
: Sum of the CPU seconds found in the first portion of the breakdown.

`Weighted seconds (total)`
: Total weighted CPU seconds; the sum of the weighted seconds found in the first portion of the breakdown.

See Example 3 of the EXAMPLES section for the multitasking breakdown, including summary statistics.

Mark and Disable Options

The mark and disable options are as follows:

`-m` Writes the position of the next accounting record to standard output. This can be used to mark various positions within the job accounting file for later use with the `-p` option. The position marked is the byte offset of the current end-of-information of the job accounting file. (`-m` cannot be used with the report selection and modifier options nor with the `-t` disable option.)

`-t` Disables (terminates) job accounting. (`-m` and `-t` are mutually exclusive).

Report Modifier Options

Report modifier options must be used with at least one selection option. The report modifier options are as follows:

`-d` Provides information about device-specific I/O, if available; forces `-l` to be selected.

`-e` Generates an extended summary report; you must use `-e` with the `-s` option. The following are descriptions of fields produced by specifying the `-e` option with the `-s` option. These fields provide additional accumulated statistics for the reporting period. Several fields contain values only if performance accounting has been enabled; otherwise, the string `NA` is printed instead.

`System Call Time` Total amount of time (in seconds) that the processes executed system calls.

`I/O Wait Time (Terminals)` Total amount of time in seconds that the processes waited for I/O from and to terminals. This field contains a significant value only if performance accounting is enabled (see `devacct`(8)).

`Wait Time while Swapped` Total amount of time (in seconds) that the processes waited while swapped out of memory. This field contains a significant value only if performance accounting is enabled (see `devacct`(8)).

`Number of Swaps` Number of times the processes were swapped out of memory.

`Physical Blocks Moved (Bufd I/O)` Number of physical blocks transferred by processes to and from block devices by using the system buffer I/O interface. This field contains a significant value only if performance accounting is enabled (see `devacct`(8)).

`Physical Blocks Moved (Raw I/O)` Number of physical blocks transferred by processes to and from block devices by using the raw I/O interface. This field contains a significant value only if performance accounting is enabled (see `devacct`(8)).

`-C` Replaces I/O wait times with connect time; must be used with the `-l` option.

`-h` Replaces physical I/O data with the largest amount of memory the process used at one time, in 512-word units. Used only with both the `-c` and `-l` options.

`-l` Provides additional information when used with `-c` or `-f`. Additionally, if the `-l` and `-c` options are used, and there is an MPP accounting record (where there is a Cray MPP system), three MPP fields are printed:

CPU time	Amount of CPU time (in seconds) used by the process for binary execution.
PEs	Number of processing elements (PEs) used.
Barrier bits	Number of barrier bits used to control process flow (in bits per second)

`-r` Raw mode, no headers are printed.

`-a` *acid* Report is for this account ID (*acid*) only.

`-g` *gid* Report is for this group ID (*gid*) only.

`-j` *jid* Report is for this job ID (*jid*) only.

`-u` *uid* Report is for this user ID (*uid*) only.

`-n` *names* Shows only commands matching names patterns that may be regular expressions, as in `ed`(1), except that a + symbol indicates one or more occurrences.

`-p` *marks* Shows only commands within the marked range. This can be a list of ranges with each list item having the following form:

`.`	First command preceding current position
`m1`	First command following mark
`m1:`	All commands between the mark and EOF
`m1:m2`	All commands between the two marks
`:m1`	All commands between BOF and the mark
`:`	All commands between BOF and EOF (default)

See the `-m` option for information on how to obtain marks.

`-D` Reports on tape daemon usage. Tape information, such as the number of bytes read and written, is available after the tape unloads. Reservation information is available after the tape device is released.

NOTES

For multitasking breakdowns with the `-c` and `-s` options, processes are considered to be multitasked if the program was multitasked and if actual execution overlap occurred.

CAUTIONS

In the UNICOS operating system, a system administrator has the option of choosing which accounting records are written to the `pacct` file. Only the base record is required. System administrators may turn off records in the `pacct` file to save disk space and the expense of keeping certain records.

If an accounting record is not turned on in the `pacct` file, a user cannot generate that record's information with the `ja` command. In order to generate the desired information with `ja`, a system administrator must turn on the necessary record(s) in the `pacct` file for a specified time period.

EXAMPLES

The following two examples show the usage of the `-m` and `-p` options with standard shell and Korn shell variables.

Example 1:

```
ja                #enable job accounting
     .
     .  (Miscellaneous commands)
     .
m1=‘ja -m‘        #mark job accounting file’s current position
     .
     .  (Commands of special interest)
     .
m2=‘ja -m‘        #mark job accounting file’s current position
     .
     .  (Miscellaneous commands)
     .
ja -cp $m1:$m2   #print command report from mark m1 to mark m2
ja -st           #print summary report for entire session and disable
                  job accounting
```

Example 2:

```
ja            #enable job accounting
      .
      .   (Miscellaneous commands)
      .
m1='ja -m'    #mark job accounting file's current position
      .
      .   (Commands of special interest)
      .
ja -cp $m1:   #print command report from mark to EOF
      .
      .   (Miscellaneous commands)
      .
ja -st        #print summary report for entire session and disable
               job accounting
```

Example 3: The following example is `ja -s` output for a multitasked process:

```
Job Accounting - Summary Report
===============================

Job Accounting File Name         : jacct
Operating System                 : sn4025 hot 8.0.2ei tja.11 CRAY C90
User Name (ID)                   : user (100)
Group Name (ID)                  : ugrp (10)
Account Name (ID)                : user (100)
Job ID                           : 8271
Report Starts                    : 07/27/94 08:51:58
Report Ends                      : 07/27/94 08:52:10
Elapsed Time                     :           12      Seconds
User CPU Time                    :            3.5826 [      3.4636] Seconds
Multitasking Breakdown

(Concurrent CPUs [Weight] * Connect seconds = CPU seconds [Weighted seconds])
 ---------------  ------    ---------------   -----------  ----------------

               1 [  1.00] *          2.1793 =      2.1793 [          2.1793]
               2 [  1.90] *          0.2464 =      0.4929 [          0.4682]
               3 [  2.70] *          0.2803 =      0.8410 [          0.7569]
               4 [  3.40] *          0.0170 =      0.0679 [          0.0577]

Concurrent CPUs [Weight] * Connect seconds = CPU seconds [Weighted seconds])
     (Avg.)       (Avg.)       (total)         (total)     (total)
 ---------------  ------    --------------   -----------  ----------------

            1.32 [  1.27] *          2.7231 =      3.5811 [          3.4622]
```

```
System CPU Time                 :            0.0812 Seconds
I/O Wait Time (Locked)          :            0.0878 Seconds
I/O Wait Time (Unlocked)        :            0.0337 Seconds
CPU Time Memory Integral        :            1.3551 Mword-seconds
SDS Time Memory Integral        :            0.0000 Mword-seconds
I/O Wait Time Memory Integral   :            0.0424 Mword-seconds
Data Transferred                :            0.0038 MWords
Maximum memory used             :            0.4844 MWords
Logical I/O Requests            :           19
Physical I/O Requests           :           12
Number of Commands              :            2
Billing Units                   :            0.0000
```

SEE ALSO

acctcom(1), at(1), crontab(1), ed(1), ps(1), sh(1)

exec(2), fork(2), listio(2), read(2), reada(2), wait(2), write(2), writea(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

devacct(8) in the *UNICOS Administrator Commands Reference Manual*, Cray Research publication SR–2022

UNICOS Resource Administration, Cray Research publication SG–2302

NAME

jobs – Displays status of jobs in the current session

SYNOPSIS

```
jobs [-l] [-n] [job_id ...]
jobs -p [-n] [job_id ...]
```

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4
AT&T extension (-n option)

DESCRIPTION

The jobs utility displays the status of jobs that were started in the current shell execution environment (see sh(1)). When jobs reports the termination status of a job, the shell removes its process ID from the list of those "known in the current shell execution environment."

The jobs utility accepts the following options and operand:

-l Lists more information about each job. This information includes the job number, current job, process group ID, state, and the command that formed the job.

-n Displays only jobs that have stopped or exited since last notified.

-p Displays only the process IDs for the process group leaders of the specified jobs.

job_id Specifies the jobs for which the status will be displayed. If no *job_id* operand is given, the status information for all jobs is displayed. See the Jobs subsection in the sh(1) man page for a description of the format of *job_id*.

By default, the jobs utility displays the status of all stopped jobs, running background jobs, and all jobs whose status has changed and have not been reported by the shell.

If you specify the -p option, the output consists of one line for each process ID:

"<*process ID*>\n"

Otherwise, if the -l option is not specified, the output is a series of lines of the form:

"[<*job-number*>] <*current*> <*state*> <*command*>\n"

The fields are as follows:

<*current*> The character + identifies the jobs that would be used as a default for the bg(1) or fg(1) utilities; you can also specify this job using the *job_id* "%+" or "%%". The character - identifies the jobs that would become the default if the current default job were to exit; you can also specify this job using the *job_id* "%-". For other jobs, this field is a <space>. At most, one job can be identified with + and at most one job can be identified with -. If any suspended job exists, the current job is a suspended job. If at least two suspended jobs exist, the previous job also is a suspended job.

<*job-number*> A number that can be used to identify the process group to the wait(1), fg(1), bg(1), and kill(1) utilities. Using these utilities, you can identify the job by prefixing the job number with %.

<*state*> One of the following strings (in the POSIX locale):

Running	Indicates that the job has not been suspended by a signal and has not exited.
Done	Indicates that the job completed and returned exit status zero.
Done (*code*)	Indicates that the job completed normally and that it exited with the specified nonzero exit status, *code*.
Stopped Stopped (SIGTSTP)	Indicates that the job was suspended by the SIGTSTP signal.
Stopped (SIGSTOP)	Indicates that the job was suspended by the SIGSTOP signal.
Stopped (SIGTTIN)	Indicates that the job was suspended by the SIGTTIN signal.
Stopped (SIGTTOU)	Indicates that the job was suspended by the SIGTTOU signal.

<*command*> Specifies the associated command that was given to the shell.

If you specify the -l option, a field that contains the process group ID is inserted before the <*state*> field.

NOTES

The jobs utility described in this man page is a built-in utility to the standard shell (sh(1)). An executable version of this utility is available in /usr/bin/jobs.

EXIT STATUS

The jobs utility exits with one of the following values:

0 Successful completion.

>0 An error occurred.

SEE ALSO

bg(1), fg(1), kill(1), sh(1), wait(1)

NAME

join – Joins specified lines of files

SYNOPSIS

join [-a *file_number* | -v *file_number*] [-e *string*] [-o *list*] [-t *char*] [-1 *field*] [-2 *field*] *file1 file2*

Obsolescent version; may not be supported in future releases:

join [-a *file_number*] [-e *string*] [-j *field*] [-j1 *field*] [-j2 *field*] [-o *list...*] [-t *char*] *file1 file2*

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The join utility joins on standard output the two relations specified by the lines of *file1* and *file2*. If *file1* or *file2* is -, standard input will be used.

file1 and *file2* must be sorted in increasing ASCII-collating sequence on the fields on which they are to be joined, usually the first in each line.

One line is in the output for each pair of lines in *file1* and *file2* that have identical join fields. The output line typically consists of the common field, the rest of the line from *file1*, and the rest of the line from *file2*.

The default input field separators are a <space>, <tab>, or <newline> character. In this case, multiple separators count as one field separator, and leading separators are ignored. The default output field separator is a <space>.

Some of the following options use argument *file_number*. This argument should be 1 or 2, referring to either *file1* or *file2*, respectively. The join utility accepts the following options:

-a *file_number*	In addition to the typical output, produces a line for each unpairable line in file *file_number*, where *file_number* is 1 or 2.
-e *string*	Replaces empty output fields with string *string*.
-j *field*	Equivalent to -1 *field* -2 *field*.
-j1 *field*	Equivalent to -1 *field*.
-j2 *field*	Equivalent to -2 *field*.

`-o` *list*	Each output line comprises the fields specified in *list*, each element of which has the form *n* . *m* (*n* is a file number, and *m* is a field number). Element zero (0) represents the join field. The common field is not printed unless specifically requested. *list* is a single command line argument. However, in the obsolete version, the argument *list* can be multiple arguments on the command line.
`-t` *char*	Uses character *char* as a separator. Every appearance of *char* in a line is significant. Character *char* is used as the field separator for both input and output.
`-v` *file_number*	Instead of the default output, produces a line only for each unpairable line in *file_number*, where *file_number* is 1 or 2. If you specify both `-v 1` and `-v 2`, all unpairable lines are output.
`-1` *field*	Join on the *field*th field of file 1. Fields start with 1.
`-2` *field*	Join on the *field*th field of file 2. Fields start with 1.
file1 file2	The names of the input files that you specify.

NOTES

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
`system`, `secadm`	Allowed to join any files. In a privileged administrator shell environment, shell-redirected I/O is not subject to file protections.
`sysadm`	Allowed to join any files subject to security label restrictions. Shell-redirected I/O is subject to security label restrictions.

If the `PRIV_SU` configuration option is enabled, the super user is allowed to join any files. Shell-redirected I/O on behalf of the super user is not subject to file protections.

EXIT STATUS

The `join` utility exits with one of the following values:

0 All input files were output successfully.

>0 An error occurred.

BUGS

With default field separation, the collating sequence is that of `sort -b`; with `-t`, the sequence is that of a plain sort.

The conventions of `join`, `sort`(1), `comm`(1), and `uniq`(1) are incongruous.

File names that are numeric may cause conflict when the `-o` option is used right before listing file names.

EXAMPLES

Example 1: The following command line joins the `passwd` file and the `group` file, matching on the numeric group ID, and outputting the login name, the group name, and the login directory. It is assumed that the files have been sorted in ASCII-collating sequence on the group ID fields.

```
join -1 4 -2 3 -o '1.1 2.1 1.6' -t: /etc/passwd /etc/group
```

Example 2: The following command line performs the identical task as the previous example, but using the obsolescent version of `join`:

```
join -j1 4 -j2 3 -o 1.1 2.1 1.6 -t: /etc/passwd /etc/group
```

SEE ALSO

`awk`(1), `comm`(1), `sort`(1), `uniq`(1)

NAME

jstat – Displays job status information

SYNOPSIS

jstat [-j *jid*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The jstat command displays information pertaining to either one or all active jobs. If the -j option is not specified, all jobs will be displayed with the following format:

```
                nproc      sds      memory    cpu
jid  owner  use lim  use lim  use lim  use lim  command
---  -----  --- ---  --- ---  --- ---  --- ---  -------
```

The fields contain the following information:

jid	Job ID.
owner	ASCII name of job owner.
nproc	Number of processes in use and limit.
sds	Number of blocks in use and limit. A block is 512 Cray words.
memory	Number of clicks in use and limit. A click is 512 Cray words.
cpu	Number of CPU seconds used and limit.
command	Name of current command in job.

**** is used to indicate no limit for all lim fields.

When the -j option is specified, all processes are displayed for the specified job with the following format:

```
pid  status  state  utime  stime  size  addr  system call  command
---  ------  -----  -----  -----  ----  ----  -----------  -------
```

The fields contain the following information:

pid	Process ID.
status	Current status (run or sleep).
state	Current state.
utime	User time in seconds.
stime	System time in seconds.

`size`	Size of process in clicks.
`addr`	Address (in decimal) of process in memory.
`system call`	Last system call made.
`command`	Process name.

NOTES

Output from `jstat` is restricted to processes running at a security label that the calling user dominates.

If this command is installed with the default privilege assignment list (PAL), a user with the `showall` privilege text is not subject to output restrictions.

If *jid* is not a number, zero is used for the *jid*.

You must have the permissions of `/unicos` set to 644 in order for this command to execute correctly. Failure to have permissions set correctly may cause `jstat` to issue the message `permission denied`.

SEE ALSO

`privtext`(1), `ps`(1)

UNICOS Basic Administration Guide for CRAY J90 Model V based Systems, Cray Research publication SG–2416

NAME

kcp – Copies remote files and directories

SYNOPSIS

kcp [-b] [-c *bufsize*] [-p] [-r] [-x] [-k *realm*] [-s *bufsize*] [-S *tos*] *sourcelist destination*

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The kcp utility copies files between machines. The *sourcelist* argument can refer to remote files, local files, or directories; arguments can consist of either absolute or relative path names.

Remote files are specified in the format *rhost*:*file*; *rhost* is a remote host name or alias (described in hosts(5)). When the login name differs from your login name on a Cray Research system, file names on a remote host are specified as *user@rhost*:*file*. If you do not specify a full path name, the path is interpreted relative to your login directory on *rhost*. A path name on a remote host can be quoted (using \, ", or ‘) so that metacharacters are interpreted remotely.

The local file name *file* may not contain a colon (:) unless it is preceded anywhere in the name by a slash (/).

The kcp utility accepts the following options:

-b Displays the buffer sizes for the copy buffer and socket buffer during a transfer.

-c *bufsize* Sets the copy buffer size. This buffer is used for reading and writing between the data socket and the source or destination file. kcp automatically chooses a copy buffer size; however, you can alter the selection. The -c option requires an argument. An argument of off turns off copy buffer sizing and the buffer defaults to the size specified by the COPYBUFSIZE #define in the tcp_config.h file. An argument of auto sets buffer sizing to automatic and has no effect relative to default operation. A numeric argument sets the buffer to that size. The letter K or k can follow a numeric buffer size to specify a multiple of 1024. The default setting is auto. A size of 0 is synonymous with auto.

-p Preserves in its copies the modification times and modes of the source files, ignoring the user file creation mode mask (see umask(1)). By default, the mode and owner of the destination file are preserved if they already existed; otherwise, the mode of the source file modified by the umask on the destination host is used.

-r Copies each subtree that is rooted at that name when any of the source files are directories. The destination must be a directory. If you specify the -r option and any of the source files are directories, kcp copies each subtree rooted at that name; in this case, the destination must be a directory.

`-x` Selects encryption of all information that is transferring between hosts. This option is not available outside the United States and Canada. This option does not work correctly.

`-k` *realm* Specifies that `kcp` must obtain tickets for the remote host in *realm* instead of in the remote host's realm as determined by `krb_realmofhost`(3K).

`-s` *bufsize* Sets the socket buffer size. This kernel buffer is for data transfer in the data socket. `kcp` automatically chooses a socket buffer size; however, you can alter the selection. The `-s` option requires an argument. An argument of `off` turns off socket buffer sizing and the buffer defaults to the default kernel socket buffer size. An argument of `auto` sets buffer sizing to automatic and has no effect relative to default operation. A numeric argument sets the buffer to that size. The letter `K` or `k` can follow a numeric buffer size to specify a multiple of 1024. The default setting is `auto`. A size of 0 is synonymous with `auto`.

`-S` *tos* Sets the IP Type-of-Service (TOS) option for the connection to the value *tos*, which can be a numeric TOS value or a symbolic TOS name that is found in the `/etc/iptos` file.

sourcelist Specifies one or more remote files, local files, or directories.

destination Specifies the destination file or directory.

The `kcp` utility does not prompt for passwords; it uses Kerberos authentication when connecting to *rhost*. Authorization is as described in `klogin`(1).

The `kcp` utility handles third party copies, in which neither source nor target files are on the current machine. Host names also can take the form *rname@rhost* rather than use the current user name on the remote host.

BUGS

When only a directory is legal, the `kcp` utility does not detect all cases in which the target of a copy is a file.

The `kcp` utility is confused by any output that is generated by commands in a `.login`, `.profile`, or `.cshrc` file on the remote host.

When the destination machine is running the 4.2BSD version of `kcp`, you might need to specify the destination user and host name as *rhost.rname*.

Kerberos is used only for the first connection of a third-party copy; the second connection uses the standard Berkeley `rcp` protocol.

The `-x` option does not work with the current MIT code.

SEE ALSO

`cp`(1), `ftp`(1B) `klogin`(1), `rlogin`(1B), `rcp`(1) (UCB version), `rsh`(1), `umask`(1)

`kerberos`(3K), `krb_realmofhost`(3K) in the *Kerberos User's Guide*, Cray Research publication SG–2409

NAME

kdestroy – Destroys Kerberos tickets

SYNOPSIS

kdestroy [-f] [-n] [-q]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The kdestroy utility destroys the user's active Kerberos authorization tickets by writing zeros to the file that contains them. If the ticket file does not exist, kdestroy displays a message accordingly.

After overwriting the file, kdestroy removes the file from the system. The utility displays a message that indicates the success or failure of the operation. If kdestroy cannot destroy the ticket file, the utility warns you by beeping your terminal.

kdestroy also invalidates all Kerberos credentials that are stored in the kernel for the user. These credentials are used for network file system (NFS) requests.

For Kerberos tickets that are obtained from a Cray Research host, you probably will want to place the kdestroy command in your .logout file, so that your tickets can be destroyed automatically when you log out.

The kdestroy utility accepts the following options:

-f Runs without displaying the status message.

-n Keeps valid the NFS credentials that are in the kernel. They remain valid until their normal expiration time. New credentials can only be obtained by executing another kinit(1) command.

-q Does not beep your terminal when it does not destroy the tickets.

BUGS

Only the tickets in the user's current ticket file are destroyed. Separate ticket files are used to hold root instance and password changing tickets. These files must be destroyed also; otherwise, all of a user's tickets must be kept in a single ticket file.

FILES

Tickets are stored in the /tmp/tkt[uid] file unless the user has set the KRBTKFILE environment variable to be another file. Then, the indicated file is used.

SEE ALSO

kinit(1), klist(1)

NAME

keylogin – Decrypts and stores a secret key

SYNOPSIS

keylogin

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The keylogin command informs the secure Remote Procedure Call (RPC) subsystem of your intent to use secure RPC. keylogin communicates with the local keyserv(8) process, allowing it to cache information it will use when performing AUTH_DES style RPC authentication.

The keylogin command prompts you for your secure RPC password, and it uses it to decrypt your secret key stored in the publickey(3R) database. After it is decrypted, your key is stored by the local key server process keyserv(8), to be used by any secure network services, such as NFS.

NOTES

The login command will not call keylogin directly by default; therefore, all users who want to use secure RPC must first explicitly run keylogin.

SEE ALSO

chkey(1), login(1)

Remote Procedure Call (RPC) Reference Manual, Cray Research publication SR–2089

keyserv(8), newkey(8) in the *UNICOS Administrator Commands Reference Manual*, Cray Research publication SR–2022

NAME

`kill` – Terminates or signals processes

SYNOPSIS

`kill -s` *signal_name pid...*
`kill -l` [*exit_status*]
`kill -v`

Obsolescent version; may not be supported in future releases:

`kill` [*-signal_name*] *pid...*
`kill` [*-signal_number*] *pid...*

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The `kill` utility sends a signal to the process(es) specified by each *pid* operand. By default, signal number 15 (`SIGTERM`) is sent to the specified process(es). This usually kills processes that do not catch or ignore the signal.

The `kill` utility accepts the following options:

`-l` Causes all values of *signal_name* supported to be written to the standard output, if no *exit_status* operand is specified. If you specify an *exit_status* operand and it has a value of the `?` shell special parameter that corresponds to a process that was terminated by a signal, the *signal_name* corresponding to the signal that terminated the process is written. If you specify an *exit_status* operand and it is the unsigned decimal value of a signal number, the *signal_name* that corresponds to that signal is written.

`-s` *signal_name* Sends signal *signal_name* to the process. *signal_name* is specified as a symbolic name without the `SIG` prefix [see `kill`(2)]. The symbolic name `0` is recognized as representing the signal value 0.

`-v` Causes all signal numbers and their associated *signal_name* to be written to the standard output.

-signal_name Equals `-s` *signal_name*.

-signal_number Specifies a nonnegative decimal integer, *signal_number*, representing the signal to be used instead of `SIGTERM`.

exit_status A decimal integer specifying a signal number or the exit status of a process terminated by a signal.

pid A decimal integer specifying a process or process group to be signaled. The process(es) selected by positive, negative, and zero values of the *pid* operand are the same as described for `kill`(2). If the first *pid* operand is negative, it should be preceded by `--` to keep it from being interpreted as an option.

The process(es) may also be specified as a job control job ID (see `sh`(1)) that identifies a background process group to be signaled. The job control job ID notation is applicable only for invocations of `kill` in the current shell execution environment.

In the obsolescent versions, if the first argument is a negative integer, the argument is interpreted as a *-signal_number* option, not as a negative *pid* operand specifying a process group.

The process number of each asynchronous process started with & is reported by the shell (unless more than one process is started in a pipeline, in which case the number of the last process in the pipeline is reported). Process numbers can also be found by using `ps`(1).

The details of the kill process are described in `kill`(2). For example, when process number 0 is specified, all processes in the process group are signaled.

The process to be killed must belong to the current user; only an appropriately authorized user can kill a process owned by another user.

NOTES

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
`system`, `secadm`	Allowed to kill any process.
`sysadm`	Allowed to kill a process owned by another user, subject to security label restrictions. Shell-redirected I/O is subject to security label restrictions.

If the `PRIV_SU` configuration option is enabled, the super user is allowed to kill any process.

The `kill` utility described on this man page is a built-in utility to the standard shell (`sh`(1)). An executable version of this utility is available in `/bin/kill`.

The `csh`(1) utility has a built-in `kill` utility with slightly different characteristics. See `csh`(1).

EXIT STATUS

The `kill` utility exits with one of the following values:

0 At least one matching process was found for each *pid* operand, and the specified signal was successfully processed for at least one matching process.

>0 An error occurred.

EXAMPLES

Example 1: Sends the software termination signal to process ID of 3228:

```
kill 3228
```

Example 2: Same as example 1, but explicitly sends the name of the signal to the process:

```
kill -s TERM 3228
```

Example 3: Determines whether a command terminated due to a signal and the name of signal that caused the command to terminate:

```
utility_name arg1 arg2
kill -l $?
```

Example 4: Sends a SIGQUIT (signal number 3) to two background processes in the current shell execution environment:

```
kill -s QUIT %1 %3
```

SEE ALSO

csh(1), ps(1), sh(1)

kill(2), signal(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

General UNICOS System Administration, Cray Research publication SG–2301

NAME

kinit – Logs in to the Kerberos authentication and authorization system

SYNOPSIS

kinit [-irvl]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The kinit utility logs in to the Kerberos authentication and authorization system. Only registered Kerberos users can use the Kerberos system. For information about registering as a Kerberos user, see the kerberos(7) man page.

When you use kinit without options, the utility prompts for your user name and the Kerberos password and tries to authenticate your login with the local Kerberos server. kinit does not send your Kerberos password across the network in the clear.

If Kerberos authenticates the login attempt, kinit retrieves your initial ticket and puts it in the ticket file that your KRBTKFILE environment variable specified. If this variable is undefined, your ticket is stored in the /tmp directory, in the tkt*uid* file; *uid* specifies your user identification number.

Be sure to use the kdestroy(1) command to destroy any active tickets before you end your login session. You can put the kdestroy(1) command in your .logout file so that your tickets can be destroyed automatically when you log out.

The kinit utility accepts the following options:

-i Directs kinit to prompt you for a Kerberos instance.

-r Directs kinit to prompt you for a Kerberos realm. This option lets you authenticate yourself with a remote Kerberos server. (Interrealm authorization is cumbersome in Kerberos version 4.)

-v (Verbose mode) Directs kinit to print the name of the Kerberos realm, and to issue a status message that indicates the success or failure of your login attempt.

-l Directs kinit to prompt you for a ticket lifetime in minutes. Because of protocol restrictions in Kerberos version 4, this value must be between 5 and 1275 minutes.

BUGS

The -r option is not fully implemented.

SEE ALSO

kdestroy(1), klist(1)

kerberos(7) (available only online)

NAME

klist – Provides list of currently held Kerberos tickets

SYNOPSIS

klist [-s | -t] [-file *name*] [-srvtab]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The klist utility prints the name of the tickets file and the identity of the principal for which the tickets are provided (as listed in the tickets file), and it lists the principal names of all Kerberos tickets currently held by the user, along with the issue and expire time for each authenticator. Principal names are listed in the form *name.instance@realm*, with the dot (.) omitted if the instance is null, and the at sign (@) omitted if the realm is null.

The klist utility accepts the following options:

-s Specifies that klist does not print the issue and expire times, the name of the tickets file, or the identity of the principal.

-t Directs klist to check for the existence of a nonexpired ticket-granting ticket in the ticket file. If one is present, it exits with status 0; otherwise, it exits with status 1. When you specify this option, no output is generated.

-file *name* Specifies *name* as the ticket file. Otherwise, if you set the KRBTKFILE environment variable, this value is used. If you do not set this environment variable, the /tmp/tkt*uid* file is used; *uid* is the current user ID of the user.

-srvtab Specifies that the file is treated as a service key file, and it prints the names of the keys that it contains. If you do not specify the file with a -file option, the default is /etc/srvtab.

BUGS

When a file is being read as a service key file, very little sanity or error checking is performed.

FILES

/etc/krb.conf	File that gets the name of the local realm
/etc/srvtab	Default service key file
/tmp/tkt*uid*	Default ticket file (*uid* is the decimal UID of the user)

SEE ALSO

kdestroy(1), kinit(1)

NAME

`klogin` – Performs remote login

SYNOPSIS

`klogin` *rhost* [`-d`] [`-e`*c*] [`-k` *realm*] [`-l` *username*] [`-x`] [`-8`] [`-E`] [`-K`] [`-L`] [`-S` *tos*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `klogin` utility connects your terminal on the current local host system to the remote host system *rhost*. It uses Kerberos authentication to determine the authorization to use a remote account. Your remote terminal type is the same as your local terminal type (as specified in your `TERM` environment variable). All echoing occurs at the remote site; therefore, the `klogin` is transparent (except for delays). Flow control through <`CONTROL-s`> and <`CONTROL-q`> and flushing interrupt input and output are handled properly.

Users can create a private authorization list in a `.klogin` file in their login directories. Each line in this file should contain a Kerberos principal name of the form *principal.instance@realm*. *principal* is either an end user or a network service offered by a host computer; *instance* further qualifies the principal. If the principal is a service, the instance specifies the name of the machine on which that service runs. If the principal is a user name that has general user privileges, the instance is usually null.

If the originating user is authenticated to one of the principals specified in `.klogin`, access is granted to the account. If no `.klogin` file exists, the principal *username@localrealm* is granted access; *localrealm* is the name of an administration entity that maintains authentication data.

For example, if `user1` wants `user2` to use `klogin` to log in to `user1`'s account, `user1` must create a `.klogin` file that contains the following:

```
user1@CRAY.COM
user2@CRAY.COM
```

Note that `user1`'s user name must be put in the file; otherwise, access to the account is denied.

If `klogin` encounters a problem obtaining the Kerberos authentication information, it prints an error message and exits.

A line of the form `~.` disconnects from the remote host; `~` is the escape character. A line of the form `~`<`CONTROL-z`> suspends the `klogin` process, and a line of the form `~`<`CONTROL-y`> suspends the send portion of the `klogin` process, but allows output from the remote system. A line of the form `~z` is the same as `~`<`CONTROL-z`>.

The `klogin` utility accepts the following options:

rhost Specifies the name of the remote host.

`-d` Uses `setsockopt`(2) to turn on socket debugging on the TCP sockets that are used for communication with the remote host.

`-e`*c* Specifies an escape character (*c*). No space can separate this option flag (`-e`) and the new escape character (*c*).

`-k` *realm* Directs `klogin` to obtain tickets for the remote host in *realm* instead of in the remote host's realm, as determined by `krb_realmofhost`(3K).

`-l` *username* Specifies the account name (*username*) to use when logging in to the remote machine. The default is the current account name.

`-x` Turns on Data Encryption Standard (DES) encryption for all data passed during the `klogin` session. This option is not available outside the United States and Canada. This option significantly reduces response time and significantly increases CPU use.

`-8` Allows the transmission of 8-bit data.

`-E` Stops any character from being recognized as an escape character. When used with the `-8` option, this provides a completely transparent connection.

`-K` The `-K` option turns off all Kerberos authentication.

`-L` This allows the `klogin` process to be run in `-opost` mode (see `stty`(1)).

`-S` *tos* Sets the IP Type-of-Service (TOS) option for the connection to the value *tos*, which can be a numeric TOS value or a symbolic TOS name found in the `/etc/iptos` file.

BUGS

More of the environment should be propagated.

FILES

`/etc/hosts` TCP/IP host name database

SEE ALSO

`login`(1), `rlogin`(1B), `rsh`(1), `stty`(1)

`setsockopt`(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

`kerberos`(3K), `krb_realmofhost`(3K), `krb_sendauth`(3K) in the *Kerberos User's Guide*, Cray Research publication SG–2409

NAME

kpasswd – Changes a user's Kerberos password

SYNOPSIS

kpasswd [-h] [-n *name*] [-i *instance*] [-r *realm*] [-u *username*[.*instance*][@*realm*]]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The kpasswd command changes a Kerberos principal's password.

kpasswd accepts the following options:

-h Prints a brief summary of the options and exits.

-n *name* Uses *name* as the principal name rather than the user name of the user running kpasswd. (*name* is determined from the ticket file, if it exists; otherwise, it is determined from the UNIX user ID.)

-i *instance* Uses *instance* as the instance rather than using a null instance.

-r *realm* Uses *realm* as the realm rather than using the local realm.

-u *username*[.*instance*][@*realm*]
Indicates a fully qualified Kerberos principal.

The command prompts for the current Kerberos password (printing the name of the principal for which it intends to change the password), which is verified by the Kerberos server. If the old password is correct, the user is prompted twice for the new password. A message is printed, indicating the success or failure of the password-changing operation.

BUGS

The kpasswd command does not handle names, instances, or realms that have special characters in them when the -n, -i, or -r options are used. If you specify the -u option, however, any valid user name is accepted.

If the principal does not exist for the password you are trying to change, you will not be told until after you have entered the old password.

The kpasswd command depends on a compatible implementation of kadmind running on the Kerberos server. Such a server is provided in the MIT Project Athena distribution of Kerberos version 4. Incompatible implementations include the kpasswdd program distributed with Berkeley 4.3-Reno.

SEE ALSO

kinit(1), passwd(1)

NAME

krsh – Connects to the remote shell

SYNOPSIS

krsh *host* [-l *username*] [-n] [-d] [-k *realm*] [-S *tos*] [*command*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The krsh utility connects to the specified host and executes the specified command. krsh copies its standard input to the remote command, copies the standard output of the remote command to its standard output, and copies the standard error of the remote command to its standard error. Interrupt, quit, and terminate signals are propagated to the remote command; krsh usually terminates when the remote command does.

The krsh utility accepts the following options:

host Specifies the name of the host to which krsh will connect.

-l *username* Specifies the remote user name to be used. If you omit this option, your local user name is used. Kerberos authentication is used, and authorization is determined as in klogin(1).

-n Redirects input from the special device /dev/null. See the BUGS section for more information on redirecting input.

-d Turns on socket debugging (through setsockopt(2)) on the TCP sockets that are used for communication with the remote host.

-k *realm* Directs krsh to obtain tickets for the remote host in *realm* instead of in the remote host's realm, as determined by the krb_realmofhost(3K) command.

-S *tos* Sets the IP Type-of-Service (TOS) option for the connection to the value *tos*, which can be a numeric TOS value or a symbolic TOS name that is found in the /etc/iptos file.

command Specifies the command to be executed. If you omit *command*, instead of executing a single command, you will be logged in on the remote host through rlogin(1B).

Shell metacharacters that are not enclosed in quotation marks are interpreted on the local machine; those that are enclosed in quotation marks are interpreted on the remote machine. Thus, in the following example, the first command appends the remote file remotefile to the local file localfile; the second command appends remotefile to otherremotefile:

```
krsh otherhost cat remotefile >> localfile

krsh otherhost cat remotefile ">>" otherremotefile
```

BUGS

If you are using csh(1) and you put a krsh utility in the background without redirecting its input away from the terminal, it will block, even if the remote command posts no read operations. If no input is desired, you should use the -n option to redirect the input of krsh to /dev/null.

You cannot run an interactive command (such as vi(1)); use klogin(1).

Stop signals stop only the local krsh process.

FILES

/etc/hosts	TCP/IP host name database

SEE ALSO

csh(1), klogin(1), login(1), rlogin(1B), vi(1)

setsockopt(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

kerberos(3K), krb_realmofhost(3K), krb_sendauth(3K) in the *Kerberos User's Guide*, Cray Research publication SG–2409

NAME

ksh, rksh, sh, rsh – Korn shell and standard shell, command and programming language

SYNOPSIS

ksh [-a] [-b] [-C] [-c *string*] [-e] [-f] [-h] [-i] [-k] [-m] [-n] [-o *option*] [-p] [-r] [-S] [-s] [-t] [-u] [-v] [-x] [*arg*]

rksh [-a] [-b] [-C] [-c *string*] [-e] [-f] [-h] [-i] [-k] [-m] [-n] [-o *option*] [-p] [-r] [-S] [-s] [-t] [-u] [-v] [-x] [*arg*]

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4
AT&T extensions (-h, -k, -p, -r, and -t options)
CRI extensions (-S option)

DESCRIPTION

sh and rsh invoke the standard shell, which is functionally equivalent to the Korn shell.

ksh is a command interpreter and programming language that executes commands read from a terminal or a file. rksh is a restricted version of the command interpreter ksh; it is used to set up login names and execution environments whose capabilities are more controlled than those of the standard shell.

Invocation

If the shell is invoked by exec(2), and the first character of argument 0 ($0) is -, the shell is assumed to be a login shell and commands are read from /etc/profile and then from either .profile in the current directory or $HOME/.profile, if either file exists. Next, commands are read from the file named by the ENV parameter and parameter substitution is performed, if the file exists. If the -s flag is not present and *arg* is present, a path search is performed on the first *arg* to determine the name of the script to execute. The *arg* script must have read permission and any setuid and setgid settings will be ignored. Commands are then read as described below; the following flags are interpreted by the shell when it is invoked:

-c *string*
If the -c flag is present, commands are read from *string*.

-s If the -s flag is present or no arguments remain, commands are read from the standard input. Shell output, except for the output of the special commands, is written to file descriptor 2.

`-i` If the `-i` flag is present or if the shell input and output are attached to a terminal (as told by `ioctl`(2)), this shell is interactive. In this case, `TERM` is ignored (so that `kill 0` does not kill an interactive shell) and `INTR` is caught and ignored (so that `wait` is interruptible). In all cases, `QUIT` is ignored by the shell.

`-r` If the `-r` flag is present, the shell is a restricted shell.

The remaining flags and arguments are described under the `set` command in the Special Commands subsection.

`rksh` Only

The `rksh` command is used to set up login names and execution environments whose capabilities are more controlled than those of the standard shell. The actions of `rksh` are identical to those of `sh`, except that the following are disallowed:

- Changing directory (see `cd`(1))
- Setting the value of `SHELL`, `ENV`, or `PATH`
- Specifying path or command names containing `/`
- Redirecting output (`>`, `>|`, `<>`, and `>>`)

These restrictions are enforced after the `.profile` and `ENV` files are interpreted.

When a command to be executed is found to be a shell procedure, `rksh` invokes `ksh` to execute it. Thus, it is possible to provide the end user with shell procedures that have access to the full power of the standard shell, while imposing a limited menu of commands; this scheme assumes that the end user does not have write and execute permissions in the same directory.

The net effect of these rules is that the writer of the `.profile` has complete control over user actions, by performing guaranteed setup actions and leaving the user in an appropriate directory (probably not the login directory).

Definitions

A metacharacter is one of the following characters:

```
;   &   (   )   |   <   >   <newline>   <space>   <tab>
```

A blank is a `<tab>` or a `<space>`. An identifier is a sequence of letters, digits, or underscores starting with a letter or underscore. Identifiers are used as names for functions and named parameters. A word is a sequence of characters separated by one or more metacharacters not enclosed in quotation marks.

A command is a sequence of characters in the syntax of the shell language. The shell reads each command and carries out the desired action either directly or by invoking separate utilities. A special command is a command that is carried out by the shell without creating a separate process. Except for documented side effects, most special commands can be implemented as separate utilities.

Commands

A simple command is a sequence of blank-separated words that may be preceded by a parameter assignment list. See the Environment subsection. The first word specifies the name of the command to be executed. Except as specified below, the remaining words are passed as arguments to the invoked command. The command name is passed as argument 0 (see `exec`(2)). The *value* of a simple command is its exit status if it terminates normally, or (octal) 200+*status* if it terminates abnormally (for a list of status values, see `signal`(2)).

A pipeline is a sequence of one or more commands separated by `|`. The standard output of each command but the last is connected by a pipe (see `pipe`(2)) to the standard input of the next command. Each command is run as a separate process; the shell waits for the last command to terminate. The exit status of a pipeline is the exit status of the last command.

A list is a sequence of one or more pipelines separated by `;`, `&`, `&&`, or `||`, and optionally terminated by `;`, `&`, or `|&`. Of these five symbols, `;`, `&`, and `|&` have equal precedence, which is lower than that of `&&` and `||`. The symbols `&&` and `||` also have equal precedence. A semicolon (`;`) causes sequential execution of the preceding pipeline; an ampersand (`&`) causes asynchronous execution of the preceding pipeline (that is, the shell does not wait for that pipeline to finish). The symbol `|&` causes asynchronous execution of the preceding command or pipeline with a two-way pipe established to the parent shell. The standard input and output of the spawned command can be written to and read from by the parent shell by using the `-p` option of the special commands `read` and `print` described later. The symbol `&&` (`||`) causes the list following it to be executed only if the preceding pipeline returns a value of zero for `&&` (nonzero for `||`). An arbitrary number of new lines may appear in a list, instead of a semicolon, to delimit a command.

A command is either a simple command or one of the following. Unless otherwise stated, the value returned by a command is that of the last simple command executed in the command.

`for` *identifier* [`in` *word* ...] `;do` *list* `;done`

Each time a `for` command is executed, *identifier* is set to the next *word* taken from the `in` *word* list. If `in` *word* ... is omitted, the `for` command executes the `do` *list* once for each positional parameter that is set (see the Parameter Substitution subsection). Execution ends when there are no more words in the list.

`select` *identifier* [`in` *word* ...] `;do` *list* `;done`

A `select` command prints on standard error (file descriptor 2), the set of *word*s, each preceded by a number. If `in` *word* ... is omitted, the positional parameters are used instead (see the Parameter Substitution subsection). The `PS3` prompt is printed and a line is read from the standard input. If this line consists of the number of one of the listed *word*s, the value of the parameter *identifier* is set to the *word* corresponding to this number. If this line is empty, the selection list is printed again. Otherwise, the value of the parameter *identifier* is set to null. The contents of the line read from standard input is saved in the parameter `REPLY`. The *list* is executed for each selection until a `break` or end-of-file is encountered.

`case` *word* `in` [[`(`]*pattern* [`|` *pattern*] ... `)` *list* `;;`] ... `esac`
A `case` command executes the *list* associated with the first *pattern* that matches *word*. The form of the patterns is the same as that used for file-name generation (see the File Name Generation subsection).

`if` *list* `;then` *list* [`elif` *list* `;then` *list*] ... [`;else` *list*] `;fi`
The *list* following `if` is executed and, if it returns a 0 exit status, the *list* following the first `then` is executed. Otherwise, the *list* following `elif` is executed and, if its value is 0, the *list* following the next `then` is executed. Failing that, the `else` *list* is executed. If no `else` *list* or `then` *list* is executed, then the `if` command returns a 0 exit status.

`while` *list* `;do` *list* `;done`
`until` *list* `;do` *list* `;done`
A `while` command repeatedly executes the `while` *list* and, if the exit status of the last command in the list is 0, executes the `do` *list*; otherwise, the loop terminates. If no commands in the `do` *list* are executed, then the `while` command returns a 0 exit status; `until` may be used in place of `while` to negate the loop termination test.

`(`*list*`)` Execute *list* in a separate environment. If two adjacent open parentheses are needed for nesting, a space must be inserted to avoid arithmetic evaluation as described below.

`{`*list*`;}` *list* is simply executed. Unlike the metacharacters `(` and `)`, `{` and `}` are reserved words and must come at the beginning of a line or after `;` in order to be recognized.

`[[`*expression*`]]`
Evaluates *expression* and returns a 0 exit status when *expression* is true. For a description of *expression*, see the Conditional Expressions subsection.

`function` *identifier* `{` *list* `;}`
identifier `()` `{` *list* `;}`
Defines a function referenced by *identifier*. The body of the function is the *list* of commands between `{` and `}`. (See the Functions subsection.)

`time` *pipeline*
The *pipeline* is executed and the elapsed time as well as the user and system time are printed on standard error. Because output redirection is set up within `ksh` after the `time` simple command is performed, standard error cannot be redirected to a file.

The following reserved words are recognized only as the first word of a command and when not enclosed in quotation marks:

```
if then else elif fi case esac for while until do done { } function select time [[ ]]
```

Comments

A word beginning with # causes that word and all the following characters up to a new line to be ignored.

Aliasing

The first word of each command is replaced by the text of an `alias` if an `alias` for this word has been defined. The first character of an `alias` name can be any nonspecial printable character, but the rest of the characters must be the same as for a valid *identifier*. The replacement string can contain any valid shell script including the metacharacters listed above. The first word of each command in the replaced text, other than any that are in the process of being replaced, will be tested for aliases. If the last character of the alias value is a blank, the word following the alias will also be checked for alias substitution. Aliases can be used to redefine special built-in commands but cannot be used to redefine the reserved words listed above. Aliases can be created, listed, and exported by using the `alias` command and can be removed by using the `unalias` command. Exported aliases remain in effect for scripts invoked by name, but they must be reinitialized for separate invocations of the shell (see the Invocation subsection).

Aliasing is performed when scripts are read, not while they are executed. Therefore, for an alias to take effect the `alias` definition command must be executed before the command that references the alias is read.

Aliases are frequently used as a short form of full path names. An option to the aliasing facility allows the value of the alias to be set automatically to the full path name of the corresponding command. These aliases are called tracked aliases. The value of a tracked alias is defined the first time the corresponding command is looked up and becomes undefined each time the `PATH` variable is reset. These aliases remain tracked so that the next subsequent reference will redefine the value. Several tracked aliases are compiled into the shell. The `-h` option of the `set` command makes each referenced command name into a tracked alias.

The following exported aliases are compiled into the shell but can be unset or redefined:

```
autoload='typeset -fu'
command='command'
false='let 0'
functions='typeset -f'
hash='alias -t -'
history='fc -l'
integer='typeset -i'
local=typeset
nohup='nohup'
r='fc -e -'
stop='kill -STOP'
suspend='kill -STOP $$'
true=':'
type='whence -v'
```

Tilde Substitution

After alias substitution is performed, each word is checked to see whether it begins with an unquoted tilde (~). If it does, then the word up to a `/` is checked to see whether it matches a user name in the `/etc/passwd` file. If a match is found, the `~` and the matched login name is replaced by the login directory of the matched user. This is called a tilde substitution. If no match is found, the original text is left unchanged. A `~` by itself, or in front of a `/`, is replaced by the value of the `HOME` parameter. A `~` followed by a `+` or `-` is replaced by `$PWD` and `$OLDPWD`, respectively.

In addition, tilde substitution is attempted when the value of a variable assignment parameter begins with a ~.

Command Substitution

The standard output from a command enclosed in parenthesis preceded by a dollar sign (`$()`) or a pair of grave accents (``) may be used as part or all of a word; trailing new lines are removed. In the second (archaic) form, the string between the grave accents is processed for special quoting characters before the command is executed. (See the Quoting subsection). The command substitution `$(cat file)` can be replaced by the equivalent but faster `$(<file)`. Command substitution of most special commands that do not perform input/output redirection are carried out without creating a separate process.

An arithmetic expression enclosed in double parentheses and preceded by a dollar sign (`$(())`) is replaced by the value of the arithmetic expression within the double parenthesis.

Parameter Substitution

A parameter is an identifier, one or more digits, or any of the characters `*`, `@`, #, `?`, `-`, `$`, and `!`. A named parameter (a parameter denoted by an identifier) has a value and `0` or more attributes. Named parameters can be assigned values and attributes by using the `typeset` special command. The attributes supported by the shell are described later with the `typeset` special command. Exported parameters pass values and attributes to the environment.

The shell supports a one-dimensional array facility. An element of an array parameter is referenced by a subscript. A subscript is denoted by a `[`, followed by an arithmetic expression (see the Arithmetic Evaluation subsection) followed by a `]`. To assign values to an array, use `set -A` *name value* . . . . The value of all subscripts must be in the range of 0 through 1023. Arrays need not be declared. Any reference to a named parameter with a valid subscript is legal and an array will be created if necessary. Referencing an array without a subscript is equivalent to referencing the element 0. Since `$array` and `${array[0]}` are equivalent, exporting `array` is equivalent to exporting `array[0]`.

The *value* of a named parameter may also be assigned as follows:

name=*value* [*name*=*value*] . . .

If the integer attribute, `-i`, is set for *name*, the *value* is subject to arithmetic evaluation as described below.

Positional parameters (parameters denoted by a number) may be assigned values with the `set` special command. Parameter `$0` is set from argument 0 when the shell is invoked.

The character `$` is used to introduce substitutable *parameters.*

${*parameter*}
The shell reads all the characters from ${ to the matching } as part of the same word, even if it contains braces or metacharacters. Substitues the value, if any, of the parameter. The braces are required when *parameter* is followed by a letter, digit, or underscore that is not to be interpreted as part of its name or when a named parameter is subscripted. If *parameter* is one or more digits, it is a positional parameter. A positional parameter of more than one digit must be enclosed in braces. If *parameter* is * or @, all the positional parameters, starting with $1, are substituted (separated by a field separator character). If an array *identifier* with subscript * or @ is used, the value for each of the elements is substituted (separated by a field separator character).

${#*parameter*}
If *parameter* is * or @, substitutes the number of positional parameters. Otherwise, substitutes the length of the value of *parameter*.

${#*identifier*[*]}
Substitutes the number of elements in the array *identifier*.

${*parameter*:-*word*}
If *parameter* is set and is nonnull, substitutes its value; otherwise, substitutes *word*.

${*parameter*:=*word*}
If *parameter* is not set or is null, sets it to *word*; then substitutes the value of the parameter. Positional parameters may not be assigned to in this way.

${*parameter*:?*word*}
If *parameter* is set and is nonnull, substitutes its value; otherwise, prints *word* and exits from the shell. If *word* is omitted, prints a standard message.

${*parameter*:+*word*}
If *parameter* is set and is nonnull, substitutes *word*; otherwise, substitutes nothing.

${*parameter*#*pattern*}
${*parameter*##*pattern*}
If the shell *pattern* matches the beginning of the value of *parameter*, the value of this substitution is the value of the *parameter* with the matched portion deleted; otherwise, substitutes the value of this *parameter*. The first form deletes the smallest matching pattern; the second form deletes the largest matching pattern.

${*parameter*%*pattern*}
${*parameter*%%*pattern*}
If the shell *pattern* matches the end of the value of *parameter*, the value of this substitution is the value of the *parameter* with the matched part deleted; otherwise, substitutes the value of *parameter*. The first form deletes the smallest matching pattern; the second form deletes the largest matching pattern.

In the preceding paragraph, *word* is not evaluated unless it is to be used as the substituted string, so that, in the following example, `pwd` is executed only if `d` is not set or is null:

```
echo ${d:-$(pwd)}
```

If the colon (`:`) is omitted from the above expressions, the shell checks only whether or not *parameter* is set.

The following parameters are set automatically by the shell:

#	The number of positional parameters in decimal.
-	Flags supplied to the shell on invocation or by the `set` command.
?	The decimal value returned by the last executed command.
$	The process number of this shell.
_	Initially, the value _ is an absolute path name of the shell or script being executed as passed in the *environment*. Subsequently it is assigned the last argument of the previous command. This parameter is not set for commands that are asynchronous. This parameter is also used to hold the name of the matching `MAIL` file when checking for mail.
!	The process number of the last background command invoked.

The following environment variables are set by the shell:

ERRNO	The value of `errno` as set by the most recently failed system call. This value is system-dependent and is intended for debugging purposes.
LINENO	The line number of the current line within the script or function being executed.
OLDPWD	The previous working directory set by the `cd`(1) command.
OPTARG	The value of the last option-argument processed by the `getopts` special command.
OPTIND	The index of the last option-argument processed by the `getopts` special command.
PPID	The process number of the parent of the shell.
PWD	The present working directory set by the `cd`(1) command.
RANDOM	Each time this parameter is referenced, a random integer uniformly distributed between 0 and 32,767 is generated. The sequence of random numbers can be initialized by assigning a numeric value to `RANDOM`.
REPLY	This parameter is set by the `select` statement and by the `read` special command when no arguments are supplied.
SECONDS	Each time this parameter is referenced, the number of seconds since shell invocation is returned. If this parameter is assigned a value, the value returned upon reference will be the value that was assigned plus the number of seconds since the assignment.

The following environment variables are used by the shell:

CDPATH The search path for the cd(1) command.

COLUMNS If this variable is set, the value is used to define the width of the edit window for the shell edit modes and for printing select lists.

EDITOR If the value of this variable ends in emacs, gmacs, or vi and the VISUAL variable is not set, the corresponding option will be turned on. (See set in the Special Commands subsection.)

ENV If this parameter is set, parameter substitution is performed on the value to generate the path name of the script that will be executed when the shell is invoked. (See the Invocation subsection.) This file is typically used for alias and function definitions.

FCEDIT The default editor name for the fc command.

FPATH The search path for function definitions. This path is searched when a function with the -u attribute is referenced and when a command is not found. If an executable file is found, it is read and executed in the current environment.

IFS Internal field separators, usually <space>, <tab>, and <newline>, that are used to separate command words that result from command or parameter substitution and for separating words with the special command read. The first character of the IFS parameter is used to separate arguments for the "$*" substitution. (See the Quoting subsection.)

HISTFILE If this parameter is set when the shell is invoked, the value is the path name of the file that will be used to store the command history. (See the Command Reentry subsection.)

HISTSIZE If this parameter is set when the shell is invoked, the number of previously entered commands that are accessible by this shell will be greater than or equal to this number. The default is 128.

HOME The default argument (home directory) for the cd(1) command.

LINES If this variable is set, the value is used to determine the column length for printing select lists. select lists will print vertically until about two-thirds of LINES lines are filled.

MAIL If this parameter is set to the name of a mail file and the MAILPATH parameter is not set, the shell informs the user of arrival of mail in the specified file.

MAILCHECK This variable specifies how often (in seconds) the shell will check for changes in the modification time of any of the files specified by the MAILPATH or MAIL parameters. The default value is 600 seconds. When the time has elapsed, the shell will check before issuing the next prompt.

MAILPATH A colon (:) -separated list of file names. If this parameter is set, the shell informs the user of any modifications to the specified files that have occurred within the last MAILCHECK seconds. Each file name can be followed by a ? and a message to be printed. The message will undergo parameter substitution with the parameter $_ defined as the name of the file that has changed. The default message is you have mail in $_.

PATH The search path for commands. (See the Execution subsection.) The user may not change PATH if executing under rksh (except in .profile).

PS1 The value of this parameter is expanded for parameter substitution to define the primary prompt string which by default is ''$ ''. The character ! in the primary prompt string is replaced by the *command* number. (See the Command Reentry subsection.)

PS2 Secondary prompt string, by default ''> ''.

PS3 Selection prompt string used within a select loop, by default ''#? ''.

PS4 The value of this parameter is expanded for parameter substitution and precedes each line of an execution trace. If omitted, the execution trace prompt is ''+ ''.

SHELL The path name of the shell is kept in the environment. At invocation, if the base name of this variable matches the pattern *r*sh, the shell becomes restricted.

TMOUT If set to a value greater than 0, the shell will terminate if a command is not entered within the prescribed number of seconds after issuing the PS1 prompt. (The shell can be compiled with a maximum bound for this value that cannot be exceeded.)

VISUAL If the value of this variable ends in emacs, gmacs, or vi, the corresponding option will be turned on. (See the Special Command set.)

The shell gives default values to PATH, PS1, PS2, MAILCHECK, TMOUT, and IFS, while HOME, SHELL, ENV, and MAIL are not set at all by the shell (although HOME is set by login(1)). On some systems MAIL and SHELL are also set by login(1)).

Blank Interpretation

After parameter and command substitution, the results of substitutions are scanned for the field-separator characters (those found in IFS) and split into distinct arguments where such characters are found. Explicit null arguments ("" or '') are retained. Implicit null arguments (those resulting from *parameters* that have no values) are removed.

File Name Generation

Following substitution, unless the -f option has been set, each command word is scanned for the characters *, ?, and [. If one of these characters appears, the word is regarded as a pattern. The word is replaced with lexicographically sorted file names that match the pattern. If no file name is found that matches the pattern, the word is left unchanged. When a pattern is used for file-name generation, the character . at the start of a file name or immediately following a /, as well as the character / itself, must be matched explicitly. In other instances of pattern matching, the / and . are not treated specially.

* Matches any string, including the null string.

? Matches any single character.

[...] Matches any one of the enclosed characters. A pair of characters separated by - matches any character lexically between the pair, inclusive. If the first character following the opening "[" is a "! ", any character not enclosed is matched. A - can be included in the character set by putting it as the first or last character.

A *pattern-list* is a list of one or more patterns separated by each other with a `|`. Composite patterns can be formed with one or more of the following:

`?(`*pattern-list*`)`	Optionally matches any one of the given patterns.
`*(`*pattern-list*`)`	Matches 0 or more occurrences of the given patterns.
`+(`*pattern-list*`)`	Matches one or more occurrences of the given patterns.
`@(`*pattern-list*`)`	Matches exactly one of the given patterns.
`!(`*pattern-list*`)`	Matches anything except one of the given patterns.

Quoting

Each of the metacharacters (see the Definitions subsection) has a special meaning to the shell and causes termination of a word unless quoted. A character may be quoted (that is, made to stand for itself) by preceding it with a `\`. The pair `\<newline>` is ignored. All characters enclosed between a pair of single quotation marks (`''`) are quoted. A single quotation mark cannot appear within single quotation marks. Inside double quotation marks (`""`), parameter and command substitution occurs and `\` quotes the characters `\`, `‘`, `"`, and `$`. The meaning of `$*` and `$@` is identical when not quoted or when used as a parameter assignment value or as a file name. However, when used as a command argument, `$*` is equivalent to `"$1`*d*`$2`*d*`..."`, where *d* is the first character of the `IFS` parameter, whereas `$@` is equivalent to `$1 $2` .... Inside grave accents (`‘‘`) `\` quotes the characters `\`, `‘`, and `$`. If the grave accents occur within double quotation marks, `\` also quotes the character `"`.

The special meaning of reserved words or aliases can be removed by quoting any character of the reserved word. The recognition of function names or special command names listed below cannot be altered by quoting them.

Arithmetic Evaluation

An ability to perform integer arithmetic is provided with the special command `let`. Evaluations are performed using long arithmetic. Constants are of the form [*base#*]*n* where *base* is a decimal number between 2 and 36 representing the arithmetic base and *n* is a number in that base. If *base* is omitted, base 10 is used.

An arithmetic expression uses the same syntax, precedence, and associativity of expression of the C language. All the integral operators, other than `++`, `--`, `?:`, and `,` are supported. Named parameters can be referenced by name within an arithmetic expression without using the parameter substitution syntax. When a named parameter is referenced, its value is evaluated as an arithmetic expression.

An internal integer representation of a named parameter can be specified with the `-i` option of the `typeset` special command. Arithmetic evaluation is performed on the value of each assignment to a named parameter with the `-i` attribute. If you do not specify an arithmetic base, the first assignment to the parameter determines the arithmetic base. This base is used when parameter substitution occurs.

Because many of the arithmetic operators require quotation marks, an alternative form of the `let` command is provided. For any command which begins with a `((`, all the characters until a matching `))` are treated as a expression enclosed in quotation marks. More precisely, `(( ... ))` is equivalent to `let "..."`.

Prompting

When used interactively, the shell prompts with the value of PS1 before reading a command. If at any time a new line is typed and further input is needed to complete a command, the secondary prompt (that is, the value of PS2) is issued.

Conditional Expressions

A conditional expression is used with the [[compound command to test attributes of files and to compare strings. Word splitting and file-name generation are not performed on the words between [[and]]. Each expression can be constructed from one or more of the following unary or binary expressions:

-a *file*	True, if *file* exists.
-b *file*	True, if *file* exists and is a block special file.
-c *file*	True, if *file* exists and is a character special file.
-d *file*	True, if *file* exists and is a directory.
-e *file*	True, if *file* exists.
-f *file*	True, if *file* exists and is an ordinary file.
-g *file*	True, if *file* exists and is has its setgid bit set.
-h *file*	True, if *file* exists and is a symbolic link.
-k *file*	True, if *file* exists and is has its sticky bit set.
-m *file*	True, if *file* exists and is migrated (type IFOFL).
-M *file*	True, if *file* exists and is migrated (has a DMF handle).
-n *string*	True, if length of *string* is nonzero.
-o *option*	True, if option named *option* is on.
-p *file*	True, if *file* exists and is a fifo special file or a pipe.
-r *file*	True, if *file* exists and is readable by current process.
-s *file*	True, if *file* exists and has size greater than 0.
-t *fildes*	True, if file descriptor number *fildes* is open and associated with a terminal device.
-u *file*	True, if *file* exists and is has its setuid bit set.
-w *file*	True, if *file* exists and is writable by current process.
-x *file*	True, if *file* exists and is executable by current process. If *file* exists and is a directory, the current process has permission to search in the directory.
-z *string*	True, if length of *string* is 0.
-L *file*	True, if *file* exists and is a symbolic link.
-O *file*	True, if *file* exists and is owned by the effective user id of this process.
-G *file*	True, if *file* exists and its group matches the effective group id of this process.
-S *file*	True, if *file* exists and is a socket.
file1 -nt *file2*	True, if *file1* exists and is newer than *file2*.
file1 -ot *file2*	True, if *file1* exists and is older than *file2*.
file1 -ef *file2*	True, if *file1* and *file2* exist and refer to the same file.
string = *pattern*	True, if *string* matches *pattern*.
string != *pattern*	True, if *string* does not match *pattern*.
string1 < *string2*,	True, if *string1* comes before *string2* based on the ASCII value of their characters.
string1 > *string2*	True, if *string1* comes after *string2* based on the ASCII value of their characters.
exp1 -eq *exp2*	True, if *exp1* is equal to *exp2*.

exp1 `-ne` *exp2*	True, if *exp1* is not equal to *exp2*.
exp1 `-lt` *exp2*	True, if *exp1* is less than *exp2*.
exp1 `-gt` *exp2*	True, if *exp1* is greater than *exp2*.
exp1 `-le` *exp2*	True, if *exp1* is less than or equal to *exp2*.
exp1 `-ge` *exp2*	True, if *exp1* is greater than or equal to *exp2*.

In each of the preceding expressions, if *file* is of the form `/dev/fd/`*n*, where *n* is an integer, the test applied to the open file whose descriptor number is *n*.

A compound expression can be constructed from these primitives by using any of the following, listed in decreasing order of precedence:

(*expression*)	True, if *expression* is true. Used to group expressions.
! *expression*	True if *expression* is false.
expression1 && *expression2*	True, if *expression1* and *expression2* are both true.
expression1 || *expression2*	True, if either *expression1* or *expression2* is true.

Input/Output

Before a command is executed, its input and output may be redirected using a special notation interpreted by the shell. The following may appear anywhere in a simple command or may precede or follow a command and they are not passed on to the invoked command. Command and parameter substitution occurs before *word* or *digit* is used except as noted. File-name generation occurs only if the pattern matches a single file and blank interpretation is not performed.

<*word*	Uses file *word* as standard input (file descriptor 0).
>*word*	Uses file *word* as standard output (file descriptor 1). If the file does not exist, it is created. If the file exists and the `noclobber` option is on, this causes an error; otherwise, it is truncated to 0 length.
>|*word*	Same as >, except that it overrides the `noclobber` option.
>>*word*	Uses file *word* as standard output. If the file exists, output is appended to it (by first seeking to the end-of-file); otherwise, the file is created.
<>*word*	Opens file *word* for reading and writing as standard input.
<<[-]*word*	The shell input is read up to a line that is the same as *word*, or to an end-of-file. No parameter substitution, command substitution, or file name generation is performed on *word*. The resulting document, called a *here-document*, becomes the standard input. If any character of *word* is enclosed in quotation marks, no interpretation is placed on the characters of the document; otherwise, parameter and command substitution occurs, `\<newline>` is ignored, and \ must be used to quote the characters \, $, ‘, and the first character of *word*. If - is appended to <<, all leading `<tab>`s are stripped from *word* and from the document.
<&*digit*	The standard input is duplicated from file descriptor *digit* (see `dup`(2)). Similarly for the standard output using >& *digit*.
<&-	The standard input is closed. Similarly for the standard output using >&-.

`<&p` The input from the co-process is moved to standard input.

`>&p` The output to the co-process is moved to standard output.

If one of the above is preceded by a digit, the file descriptor number referred to is that specified by the digit (instead of the default `0` or `1`). For example:

```
... 2>&1
```

means that file descriptor 2 is to be opened for writing as a duplicate of file descriptor 1.

The order in which redirections are specified is significant. The shell evaluates each redirection in terms of the (*file descriptor*, *file*) association at the time of evaluation. For example:

```
... 1>fname 2>&1
```

first associates file descriptor 1 with file *fname*. It then associates file descriptor 2 with the file associated with file descriptor 1 (that is, *fname*). If the order of redirections were reversed, file descriptor 2 would be associated with the terminal (assuming file descriptor 1 had been) and then file descriptor 1 would be associated with file *fname*.

If a command is followed by `&` and job control is not active, the default standard input for the command is the empty file `/dev/null`. Otherwise, the environment for the execution of a command contains the file descriptors of the invoking shell as modified by input/output specifications.

Environment

The environment (see `environ`(7)) is a list of name-value pairs that is passed to an executed program in the same way as a normal argument list. The names must be identifiers and the values are character strings. The shell interacts with the environment in several ways. On invocation, the shell scans the environment and creates a parameter for each name found, giving it the corresponding value and marking it export. Executed commands inherit the environment. If the user modifies the values of these parameters or creates new ones, using the `export` or `typeset -x` commands, the commands become part of the environment. The environment seen by any executed command is thus composed of any name-value pairs originally inherited by the shell, whose values may be modified by the current shell, plus any additions that must be noted in `export` or `typeset -x` commands.

The environment for any simple command or function may be augmented by prefixing it with one or more parameter assignments. A parameter assignment argument is a word of the form *identifier=value*. As far as the above execution of `cmd` is concerned, the following two lines are equivalent:

```
TERM=450 cmd args
(export TERM; TERM=450; cmd args)
```

If the `-k` flag is set, all parameter assignment arguments are placed in the environment, even if they occur after the command name. The following first prints `a=b c` and then `c:`

```
echo a=b c
set -k
echo a=b c
```

This feature is intended for use with scripts written for early versions of the shell, and its use in new scripts is strongly discouraged, because support will be discontinued in future releases.

Functions

The `function` reserved word, described in the Commands subsection, is used to define shell functions. Shell functions are read in and stored internally. Alias names are resolved when the function is read. Functions are executed like commands with the arguments passed as positional parameters (see the Execution subsection).

Functions execute in the same process as does the caller and share all files and present working directory with the caller. Traps caught by the caller are reset to their default action inside the function. A trap condition that is not caught or ignored by the function causes the function to terminate and the condition to be passed on to the caller. A trap on `EXIT` set inside a function is executed after the function completes in the environment of the caller. Ordinarily, variables are shared between the calling program and the function. However, the `typeset` special command used within a function defines local variables whose scope includes the current function and all functions it calls.

The special command `return` is used to return from function calls. Errors within functions return control to the caller.

Function identifiers can be listed with the `-f` or `+f` option of the `typeset` special command. The text of functions will also be listed with `-f`. Function can be undefined with the `-f` option of the `unset` special command.

Ordinarily, functions are unset when the shell executes a shell script. The `-xf` option of the `typeset` command allows a function to be exported to scripts that are executed without a separate invocation of the shell. Functions that need to be defined across separate invocations of the shell should be specified in the `ENV` file with the `-xf` option of `typeset`.

Jobs

If the `monitor` option of the `set` command is turned on, an interactive shell associates a job with each pipeline. It keeps a table of current jobs, printed by the `jobs` command, and assigns them small integer numbers. When a job is started asynchronously with &, the shell prints a line that looks as follows:

```
[1] 1234
```

This indicates that the job that was started asynchronously was job number 1 and had one (top-level) process, whose process ID was `1234`.

If you are running a job and want to do something else, you may press `<^z>` (`<CONTROL-z>`), which sends a `STOP` signal to the current job. The shell will then usually indicate that the job has been stopped, and print another prompt. You can then manipulate the state of this job, putting it in the background with the `bg` command, or run some other commands and then eventually bring the job back into the foreground with the foreground command `fg`. The consequence of pressing `<^z>` takes effect immediately and is like an interrupt in that pending output and unread input are discarded when it is typed.

A job being run in the background will stop if it tries to read from the terminal. Background jobs are usually allowed to produce output, but this can be disabled by entering the command `stty tostop`. If you set this `tty` option, background jobs will stop when they try to produce output like they do when they try to read input.

There are several ways to refer to jobs in the shell. A job can be referred to by the process ID of any process of the job or by one of the following:

%*number*	The job with the given number
%*string*	Any job whose command line begins with *string*
%?*string*	Any job whose command line contains *string*
%%	Current job
%+	Equivalent to %%
%-	Previous job

This shell learns immediately whenever a process changes state. It usually informs you whenever a job becomes blocked so that no further progress is possible, but only just before it prints a prompt. This is done so that it does not otherwise disturb your work.

When the monitor mode is on, each background job that completes triggers any trap set for `CHLD`.

If you try to leave the shell while jobs are running or stopped, you will be warned that `You have stopped(running) jobs.` You may use the `jobs` command to see what they are. If you do this or immediately try to exit again, the shell will not warn you a second time, and the stopped jobs will be terminated.

Signals

The `INT` and `QUIT` signals for an invoked command are ignored if the command is followed by & and job `monitor` option is not active. Otherwise, signals have the values inherited by the shell from its parent (but see also the `trap` command).

Execution

Each time a command is executed, the above substitutions are carried out. If the command name matches one of the commands listed in the Special Commands subsection, it is executed within the current shell process. Next, the command name is checked to see whether or not it matches one of the user-defined functions. If it matches, the positional parameters are saved and then reset to the arguments of the `function` call. When the function completes or issues a `return`, the positional parameter list is restored and any trap set on `EXIT` within the function is executed. The value of a function is the value of the last command executed. A function is also executed in the current shell process. If a command name is not a special command or a user-defined function, a process is created and an attempt is made to execute the command by using `exec`(2).

The shell parameter PATH defines the search path for the directory containing the command. Alternative directory names are separated by a colon (:). The default path is /bin:/usr/bin:/usr/ucb (specifying /bin, /usr/bin, /usr/ucb, and the current directory, in that order). The current directory can be specified by two or more adjacent colons, or by a colon at the beginning or end of the path list. If the command name contains a /, the search path is not used. Otherwise, each directory in the path is searched for an executable file. If the file has execute permission but is not a directory or an a.out file, or if the file does not begin with characters #! (see exec(2)), it is assumed to be a file containing shell commands. A subshell is spawned to read it. All nonexported aliases, functions, and named parameters are removed in this case. The shell command file must have read permission and any setuid and setgid settings will be ignored. A parenthesized command is executed in a subshell without removing nonexported quantities.

Command Reentry

The text of the last HISTSIZE (default 128) commands entered from a terminal device is saved in a history file. The file $HOME/.sh_history is used if the HISTFILE variable is not set or is not writable. A shell can access the commands of all interactive shells that use the same named HISTFILE. The special command fc is used to list or edit a portion of this file. The portion of the file to be edited or listed can be selected by number, or you can specify the first character or characters of the command. A single command or range of commands can be specified. If you do not specify an editor program as an argument to fc, the value of the parameter FCEDIT is used. If FCEDIT is not defined, /bin/ed is used. The edited command(s) is printed and reexecuted upon leaving the editor. The editor name - is used to skip the editing phase and to reexecute the command. In this case, a substitution parameter of the form *old=new* can be used to modify the command before execution. For example, if r is aliased to 'fc -e -' then typing r bad=good c will reexecute the most recent command that starts with the letter c, replacing the first occurrence of the string bad with the string good.

Inline Editing Options

Usually, each command line entered from a terminal device is simply typed followed by a newline (<RETURN> or <LINE FEED>). If either the emacs, gmacs, or vi option is active, the user can edit the command line. To be in either of these edit modes, set the corresponding option. An editing option is automatically selected each time the VISUAL or EDITOR variable is assigned a value ending in either of these option names.

The editing features require that the user's terminal accept <RETURN> as carriage return without line feed and that a space ( ) must overwrite the current character on the screen. ADM terminal users should set the "space - advance" switch to 'space'. Hewlett-Packard series 2621 terminal users should set the straps to bcGHxZ etX.

The editing modes implement a concept in which the user is looking through a window at the current line. The window width is the value of COLUMNS, if it is defined; otherwise the width is 80. The window height is set at the value of LINES, if it is defined; otherwise the height is 24. If the line is longer than the window width minus two, a mark is displayed at the end of the window to notify the user. As the cursor moves and reaches the window boundaries, the window will be centered about the cursor. The mark is a > (<, *) if the line extends on the right (left, both) side(s) of the window.

The search commands in each edit mode provide access to the history file. Only strings are matched, not patterns, although a leading ^ in the string restricts the match to begin at the first character in the line.

Emacs Editing Mode

This mode is entered by enabling either the emacs or gmacs option. The only difference between these two modes is the way they handle <^T>. To edit, the user moves the cursor to the point needing correction and then inserts or deletes characters or words as needed. All the editing commands are control characters or escape sequences. The notation for control characters is caret (^) followed by the character. For example, <^F> is the notation for pressing <CONTROL-f>. This is entered by pressing the f key while holding down the <CONTROL> key. The <SHIFT> key is not pressed. (The notation ^? indicates the <DEL> (delete) key.)

The notation for escape sequences is M- followed by a character. For example, M-f (pronounced Meta f) is entered by depressing ESC (ASCII 033) followed by f. (M-F would be the notation for ESC followed by SHIFT (capital) F.)

All edit commands operate from anyplace on the line (not just the beginning). Neither the <RETURN> nor the <LINE FEED> key is pressed after edit commands except when noted.

^F	Moves cursor forward (right) 1 character.
M-f	Moves cursor forward 1 word. (To the emacs editor, a *word* is a string of characters consisting of only letters, digits, and underscores.)
^B	Moves cursor backward (left) 1 character.
M-b	Moves cursor backward 1 word.
^A	Moves cursor to start-of-line.
^E	Moves cursor to end-of-line.
^]*char*	Moves cursor forward to character *char* on current line.
M-^]*char*	Moves cursor back to character *char* on current line.
^X^X	Interchanges cursor and mark.
erase	(User-defined erase character as defined by the stty(1) command, usually ^H or #.) Deletes previous character.
^D	Deletes current character.
M-d	Deletes current word.
M-^H	(<Meta-BACKSPACE>) Deletes previous word.
M-h	Deletes previous word.
M-^?	(<Meta-DEL>) Deletes previous word. (If your interrupt character is ^? (<DEL>, the default), this command does not work.)
^T	Transposes current character with next character in emacs mode. Transposes two previous characters in gmacs mode.

^C	Capitalizes current character.
M-c	Capitalizes current word.
M-l	Changes the current word to lowercase.
^K	Deletes from cursor to end-of-line. If preceded by a numerical parameter whose value is less than the current cursor position, deletes from given position up to the cursor. If preceded by a numerical parameter whose value is greater than the current cursor position, deletes from cursor up to given cursor position.
^W	Kills from the cursor to the mark.
M-p	Pushes the region from the cursor to the mark on the stack.
kill	(User-defined kill character as defined by the stty command, usually ^G or @.) Kills the entire current line. If two *kill* characters are entered in succession, all kill characters from then on cause a line feed (useful when using paper terminals).
^Y	Restores last item removed from line. (Yanks item back to the line.)
^L	Line feed and prints current line.
^@	(Null character) Sets mark.
M-<*space*>	(<Meta SPACE>) Sets mark.
^J	(New line) Executes the current line.
^M	(<RETURN>) Executes the current line.
eof	End-of-file character, usually ^D, is processed as an end-of-file only if the current line is null.
^P	Fetches previous command. Each time ^P is entered, the previous command back in time is accessed. Moves back one line when not on the first line of a multiline command.
M-<	Fetches the least recent (oldest) history line.
M->	Fetches the most recent (youngest) history line.
^N	Fetches next command line. Each time ^N is entered, the next command line forward in time is accessed.
^R*string*	Reverses search history for a previous command line containing *string*. If a parameter of 0 is given, the search is forward. *String* is terminated by a RETURN or NEW LINE. If string is preceded by a ^, the matched line must begin with *string*. If *string* is omitted, the next command line containing the most recent *string* is accessed. In this case a parameter of 0 reverses the direction of the search.
^O	(Operates) Executes the current line and fetches the next line relative to current line from the history file.
M-*digits*	(Escape) Defines numeric parameter; the digits are taken as a parameter to the next command. The commands that accept a parameter are ^F, ^B, *erase*, ^C, ^D, ^K, ^R, ^P, ^N, ^], M-., M-^], M-_, M-b, M-c, M-d, M-f, M-h, M-l, and M-^H.

M-*letter*	(Soft-key) Your alias list is searched for an alias by the name _*letter* and if an alias of this name is defined, its value will be inserted on the input queue. The *letter* must not be one of the previous meta-functions.
M-[*letter*	(Soft-key) Your alias list is searched for an alias by the name __*letter* and if an alias of this name is defined, its value will be inserted on the input queue. This can be used to program functions keys on many terminals.
M-.	Inserts the last word of the previous command is inserted on the line. If preceded by a numeric parameter, the value of this parameter determines which word to insert rather than the last word.
M-_	Same as M-..
M-*	Attempts file-name generation on the current word. An asterisk is appended if the word does not match any file or contain any special pattern characters.
M-ESC	File name completion. Replaces the current word with the longest common prefix of all filenames matching the current word with an asterisk appended. If the match is unique, a / is appended if the file is a directory and a space is appended if the file is not a directory.
M-=	Lists files matching current word pattern if an asterisk were appended.
^U	Multiplies parameter of next command by 4.
\	Escapes next character. Editing characters, the user's erase, kill and interrupt (usually ^?) characters may be entered in a command line or in a search string if preceded by a \. The \ removes the next character's editing features (if any).
^V	Displays version of the shell.
M-#	Inserts a # at the beginning of the line and executes it. This causes a comment to be inserted in the history file.

The vi Editing Mode

The vi editor has two typing modes. When you enter a command, you are in input mode. To edit, you enter control mode by typing ESC (033), move the cursor to the point needing correction, and then insert or delete characters or words as needed. Most control commands accept an optional repeat count prior to the command.

In vi mode on most systems, canonical processing is initially enabled and the command will be echoed again if the speed is 1200 baud or greater and it contains any control characters or less than one second has elapsed since the prompt was printed. The ESC character terminates canonical processing for the remainder of the command, and the user can then modify the command line. This scheme has the advantages of canonical processing with the type-ahead echoing of raw mode.

If the option `viraw` is also set, the terminal will always have canonical processing disabled. This mode may be helpful for certain terminals and is implicit for systems that do not support two alternative end-of-line delimiters (as defined by RFC-1184). If the shell can determine that this functionality is supported, the `viraw` option will be off by default. Otherwise, the `viraw` option will be on by default. If the shell can determine that this functionality is not supported, efforts to turn the `viraw` option off will be silently ignored.

Input Edit Commands

By default, the editor is in input mode.

erase	(User-defined erase character as defined by the `stty` command, usually `^H` or #.) Deletes previous character.
`^W`	Deletes the previous blank separated word.
`^D`	Terminates the shell.
`^V`	Escapes next character. Editing characters, the user's erase or kill characters may be entered in a command line or in a search string if preceded by a `^V`. The `^V` removes the next character's editing features (if any).
`\`	Escapes the next erase or kill character.

Motion Edit Commands

The following commands move the cursor:

[*count*]`l`	Moves cursor forward (right) 1 character.
[*count*]`w`	Moves cursor forward 1 alphanumeric word.
[*count*]`W`	Moves cursor to the beginning of the next word that follows a blank.
[*count*]`e`	Moves cursor to end-of-word.
[*count*]`E`	Moves cursor to end of the current blank delimited word.
[*count*]`h`	Moves cursor backward (left) 1 character.
[*count*]`b`	Moves cursor backward 1 word.
[*count*]`B`	Moves cursor to preceding blank separated word.
[*count*]`|`	Moves cursor to column *count*.
[*count*]`f`*c*	Finds next character *c* in the current line.
[*count*]`F`*c*	Finds previous character *c* in the current line.
[*count*]`t`*c*	Equivalent to `f` followed by `h`.
[*count*]`T`*c*	Equivalent to `F` followed by `l`.
[*count*]`;`	Repeats *count* times, the last single character find command, `f`, `F`, `t`, or `T`.
[*count*]`,`	Reverses the last single character find command *count* times.
`0`	Moves cursor to start of line.
`^`	Moves cursor to first nonblank character in line.
`$`	Moves cursor to end-of-line.

Search Edit Commands

The following commands access your command history:

[*count*]k	Fetches previous command. Each time k is entered, the previous command back in time is accessed.
[*count*]-	Equivalent to k.
[*count*]j	Fetches next command. Each time j is entered, the next command forward in time is accessed.
[*count*]+	Equivalent to j.
[*count*]G	Fetches the command number *count*. The default is the least recent history command.
/*string*	Searches backward through history for a previous command containing *string*. *string* is terminated by a <RETURN> or <NEWLINE>. If *string* is preceded by a ^, the matched line must begin with *string*. If *string* is null, the previous string will be used.
?*string*	Same as / except that search will be in the forward direction.
n	Searches for next match of the last pattern to / or ? commands.
N	Searches for next match of the last pattern to / or ?, but in reverse direction. Searches history for the string entered by the previous / command.

Text Modification Edit Commands

The following commands modify the line:

a	Enters input mode and enters text after the current character.
A	Appends text to the end of the line. Equivalent to $a.
[*count*]c*motion* c[*count*]*motion*	Deletes current character through the character to which *motion* would move the cursor and enter input mode. If *motion* is c, the entire line will be deleted and input mode entered.
C	Deletes the current character through the end-of-line and enter input mode. Equivalent to c$.
S	Equivalent to cc.
D	Deletes the current character through the end-of-line. Equivalent to d$.
[*count*]d*motion* d[*count*]*motion*	Deletes current character through the character to which *motion* would move. If *motion* is d, the entire line will be deleted.
i	Enters input mode and inserts text before the current character.
I	Inserts text before the beginning of the line. Equivalent to 0i.
[*count*]P	Places the previous text modification before the cursor.

[*count*]p	Places the previous text modification after the cursor.
R	Enters input mode and replaces characters on the screen with characters you type overlay fashion.
[*count*]r*c*	Replaces the *count* character(s) starting at the current cursor position with *c*, and advances the cursor.
[*count*]x	Deletes current character.
[*count*]X	Deletes preceding character.
[*count*].	Repeats the previous text modification command.
[*count*]~	Inverts the case of the *count* character(s) starting at the current cursor position and advances the cursor.
[*count*]_	Causes the *count* word of the previous command to be appended and input mode entered. The last word is used if *count* is omitted.
*	Causes an * to be appended to the current word and file-name generation attempted. If no match is found, it rings the bell. Otherwise, the word is replaced by the matching pattern and input mode is entered.
\	File name completion. Replaces the current word with the longest common prefix of all file names matching the current word with an asterisk appended. If the match is unique, a / is appended if the file is a directory and a space is appended if the file is not a directory.

Other Edit Commands

Miscellaneous commands.

[*count*]y*motion* y[*count*]*motion*	Yanks current character through character that *motion* would move the cursor to and puts them into the delete buffer. The text and cursor are unchanged.
Y	Yanks from current position to end-of-line. Equivalent to y$.
u	Undoes the last text modifying command.
U	Undoes all the text modifying commands performed on the line.
[*count*]v	Returns the command `fc -e ${VISUAL:-${EDITOR:-vi}}` *count* in the input buffer. If *count* is omitted, the current line is used.
^L	Line feed and prints current line. Has effect only in control mode.
^J	(<NEWLINE>) Executes the current line, regardless of mode.
^M	(<RETURN>) Executes the current line, regardless of mode.
#	Sends the line after inserting a # in front of the line. Useful for causing the current line to be inserted in the history without being executed.
=	Lists the file names that match the current word if an asterisk were appended to it.

@letter Your alias list is searched for an alias by the name *_letter* and if an alias of this name is defined, its value will be inserted on the input queue for processing.

Special Commands

The following simple commands are executed in the shell process. Input/output redirection is permitted. Unless otherwise indicated, the output is written on file descriptor 1 and the exit status, when there is no syntax error, is 0. Commands that are preceded by one or two † are treated specially in the following ways:

1. Parameter assignment lists preceding the command remain in effect when the command completes.
2. I/O redirections are processed after parameter assignments.
3. Errors cause a script that contains them to abort.
4. Words, following a command preceded by †† that are in the format of a parameter assignment, are expanded with the same rules as a parameter assignment. This means that tilde substitution is performed after the = sign and word splitting and file name generation are not performed.

The special commands are as follows:

† `:` [*arg* ...] The command only expands parameters.

† `.` *file* [*arg* ...] Reads the complete *file* and then executes the commands. The commands are executed in the current shell environment. The search path specified by `PATH` is used to find the directory containing *file*. If any arguments *arg* are given, they become the positional parameters. Otherwise, the positional parameters are unchanged. The exit status is the exit status of the last command executed.

†† `alias` [`-tx`] [*name*[=*value*]] ...
Alias with no arguments prints the list of aliases in the form *name=value* on standard output. An alias is defined for each name whose *value* is given. A trailing space in *value* causes the next word to be checked for alias substitution. The `-t` flag is used to set and list tracked aliases. The value of a tracked alias is the full path name corresponding to the given *name*. The value becomes undefined when the value of `PATH` is reset but the aliases remained tracked. Without the `-t` flag, for each *name* in the argument list for which no *value* is given, the name and value of the alias is printed. The `-x` flag is used to set or print exported aliases. An exported alias is defined for scripts invoked by name. The exit status is nonzero if a *name* is given, but no value, for which no alias has been defined. See `alias`(1).

`bg` [*job*...] This command is only on systems that support job control. Puts each specified *job* into the background. The current job is put in the background if *job* is not specified. See the Jobs subsection for a description of the format of `job`. See `bg`(1).

† `break` [*n*] Exits from the enclosing `for`, `while`, `until`, or `select` loop, if any. If *n* is specified, it breaks *n* levels.

`cd` [*arg*]
`cd` *old new*
: This command can be in either of two forms. In the first form, it changes the current directory to *arg*. If *arg* is `-` , the directory is changed to the previous directory. The shell parameter `HOME` is the default *arg*. The parameter `PWD` is set to the current directory. The shell parameter `CDPATH` defines the search path for the directory containing *arg*. Alternative directory names are separated by a colon (`:`). The default path is `<null>` (specifying the current directory). The current directory is specified by a null path name, which can appear immediately after the equal sign or between the colon delimiters anywhere else in the path list. If *arg* begins with a `/`, the search path is not used. Otherwise, each directory in the path is searched for *arg*.

  The second form of `cd` substitutes the string *new* for the string *old* in the current directory name, `PWD`, and tries to change to this new directory.

  The `cd` command may not be executed by `rksh`. See `cd`(1).

`command` [`-pVv`] *command_name* [*argument* ...]
: See `command`(1) for information on using this command.

† `continue` [*n*]
: Resumes the next iteration of the enclosing `for`, `while`, `until`, or `select` loop. If *n* is specified, it resumes at the *n*th enclosing loop.

`dmmode` *n*
: Sets the data migration recall mode to *n*. For usage and description, see `dmmode`(1).

`echo` [*arg* ...]
: For usage and description, see `echo`(1).

† `eval` [*arg* ...]
: The arguments are read as input to the shell and the resulting command(s) executed.

† `exec` [*arg* ...]
: If *arg* is given, the command specified by the arguments is executed in place of this shell without creating a new process. Input/output arguments may appear and affect the current process. If no arguments are given, the effect of this command is to modify file descriptors as prescribed by the input/output redirection list. In this case, any file descriptor numbers greater than 2 that are opened with this mechanism are closed when invoking another program.

† `exit` [*n*]
: Causes the shell to exit with the exit status specified by *n*. If *n* is omitted, the exit status is that of the last command executed. An end-of-file will also cause the shell to exit except for a shell that has the `ignoreeof` option (see `set`) turned on.

†† `export` [*name*[=*value*]] ...
: The specified *name*s are marked for automatic export to the `environment` of subsequently executed commands.

`fc` [`-e` *ename*] [`-nlr`] [*first* [*last*]]

`fc -e -` [*old=new*] [*command*]

In the first form, a range of commands from *first* to *last* is selected from the last `HISTSIZE` commands that were typed at the terminal. The arguments *first* and *last* may be specified as a number or as a string. A string is used to locate the most recent command starting with the given string. A negative number is used as an offset to the current command number. If the flag `-l` is selected, the commands are listed on standard output. Otherwise, the editor program *ename* is invoked on a file containing these keyboard commands. If *ename* is not supplied, the value of the parameter `FCEDIT` (default `/bin/ed`) is used as the editor. When editing is complete, the edited command(s) is executed. If *last* is not specified, it will be set to *first*. If *first* is not specified, the default is the previous command for editing and –16 for listing. The flag `-r` reverses the order of the commands and the flag `-n` suppresses command numbers when listing.

In the second form, the *command* is reexecuted after the substitution *old=new* is performed. See `fc`(1).

`fg` [*job...*] This command is only on systems that support job control. Each *job* specified is brought to the foreground. Otherwise, the current job is brought into the foreground. For a description of the format of `job`, see the Jobs subsection. See `fg`(1).

`getopts` *optstring name* [*arg* ...]

Checks *arg* for legal options. If *arg* is omitted, the positional parameters are used. An option-argument begins with a `+` or a `-`. An option not beginning with `+` or `-` or the argument `--` ends the options. *optstring* contains the letters that `getopts` recognizes. If a letter is followed by a `:`, that option is expected to have an argument. The options can be separated from the argument by blanks.

`getopts` places the next option letter it finds inside variable *name* each time it is invoked with a `+` prepended when *arg* begins with a `+`. The index of the next *arg* is stored in `OPTIND`. The option-argument, if any, is stored in `OPTARG`.

A leading `:` in *optstring* causes `getopts` to store the letter of an invalid option in `OPTARG`, and to set *name* to `?` for an unknown option and to `:` when a required option is missing. Otherwise, `getopts` prints an error message. The exit status is nonzero when there are no more options. See `getopts`(1).

`jobs` [`-lnp`] [*job* ...]

Lists information about each given job, or all active jobs if *job* is omitted. The `-l` flag lists process IDs in addition to the normal information. The `-n` flag displays only jobs that have stopped or exited since last notified. The `-p` flag causes only the process group to be listed. For a description of the format of *job*, see the Jobs subsection. See `jobs`(1).

`kill -s` *signal_name pid...*
`kill -l` [*exit_status*]
`kill -v`

`kill` [-*signal_name*] *pid...*
`kill` [-*signal_number*] *pid...*
For usage and description, see `kill`(1).

`let` *arg* ... Each *arg* is a separate arithmetic expression to be evaluated. For a description of arithmetic expression evaluation, see the Arithmetic Evaluation subsection.

If the value of the last expression is nonzero, the exit status is `0`; otherwise it is `1`.

`print` [`-Rnprsu`[*n*]] [*arg* ...]
The shell output mechanism. With no flags or with flag `-` or `--`, the arguments are printed on standard output as described by `echo`(1). In raw mode, `-R` or `-r`, the escape conventions of `echo` are ignored. The `-R` option will print all subsequent arguments and options other than `-n`. The `-p` option causes the arguments to be written onto the pipe of the process spawned with `|&` instead of standard output. The `-s` option causes the arguments to be written onto the history file instead of standard output. The `-u` flag can be used to specify a one-digit file descriptor unit number `n` on which the output will be placed. The default is 1. If the flag `-n` is used, no `<newline>` is added to the output.

`pwd` Prints the working directory. Equivalent to `print -r - $PWD`

`read` [`-prsu`[*n*]] [*name*`?`*prompt*] [*name* ...]
The shell input mechanism. One line is read and is broken up into fields, using the characters in `IFS` as separators. In raw mode, `-r`, a `\` at the end of a line does not signify line continuation. The first field is assigned to the first *name*, the second field to the second *name*, and so on, with leftover fields assigned to the last *name*. The `-p` option causes the input line to be taken from the input pipe of a process spawned by the shell using `|&`. If the `-s` flag is present, the input will be saved as a command in the history file. The `-u` flag can be used to specify a one-digit file descriptor unit from which to read. The file descriptor can be opened with the `exec` special command. The default value of *n* is 0. If *name* is omitted, `REPLY` is used as the default *name*. The exit status is 0, unless an end-of-file is encountered. See `read`(1). An end-of-file with the `-p` option causes cleanup for this process so that another can be spawned. If the first argument contains a `?`, the remainder of this word is used as a *prompt* on standard error when the shell is interactive. The exit status is 0, unless an end-of-file is encountered.

†† `readonly` [*name*[`=`*value*]] ...
The specified *names* are marked read only, and these names cannot be changed by subsequent assignment.

† `return` [*n*] Causes a shell *function* to return to the invoking script with the return status specified by *n*. If *n* is omitted, the return status is that of the last command executed. If `return` is invoked while not in a *function* or a `.` script, it is the same as an `exit`.

`set` [±`abCefhkmnopsStuvx`] [±`o` *option*]... [±`A` *name*] [*arg* ...]
Sets options for the shell. The flags for this command have the following meanings:

`-A` Array assignment. Unsets the variable *name* and assigns values sequentially from the list *arg*. If `+A` is used, the variable *name* is not unset first.

`-a` Automatically exports all subsequent defined parameters.

`-b` Causes the shell to notify the user asynchronously of background job completion.

`-C` Prevents redirection `>` from truncating existing files. Requires `>|` to truncate a file when turned on.

`-e` Executes the `ERR` trap, if set, and exits. Used with a command that has a nonzero exit status. This mode is disabled while reading profiles.

`-f` Disables file name generation.

`-h` Each command becomes a tracked alias when first encountered.

`-k` Places all parameter assignment arguments in the command environment. All parameter assignment arguments are placed in the environment for a command, not just those that precede the command name.

`-m` Runs background jobs in a separate process group. A line will print upon completion. The exit status of background jobs is reported in a completion message. On systems with job control, this flag is turned on automatically for interactive shells.

`-n` Reads commands and checks them for syntax errors, but does not execute them. Ignored for interactive shells.

`-o` The following argument can be one of the following option names:

`allexport`	Same as `-a`.
`errexit`	Same as `-e`.
`bgnice`	All background jobs are run at a lower priority. This is the default mode.
`emacs`	Puts you in an `emacs`-style inline editor for command entry.
`gmacs`	Puts you in a `gmacs`-style inline editor for command entry.
`ignoreeof`	The shell will not exit on end-of-file. The command `exit` must be used.
`keyword`	Same as `-k`.
`markdirs`	All directory names resulting from file name generation have a trailing `/` appended.
`monitor`	Same as `-m`.
`noclobber`	Same as `-C`.
`noexec`	Same as `-n`.
`noglob`	Same as `-f`.
`nolog`	Does not save function definitions in history file.
`notify`	Same as `-b`.
`nounset`	Same as `-u`.

`privileged` Same as `-p`.
`verbose` Same as `-v`.
`trackall` Same as `-h`.
`vi` Puts you in insert mode of a `vi`-style inline editor. Continues until you hit escape character (`033`). This puts you in move mode. A return sends the line.
`viraw` Processes each character as it is typed in `vi` mode.
`xtrace` Same as `-x`.

For `ksh -o`, the *option* argument is required. For `set -o`, if no option name is supplied, the current option settings are printed.

`-p` Disables processing of the `$HOME/.profile` file. Uses the file `/etc/suid_profile` instead of the `ENV` file. This mode is on whenever the effective uid (gid) is not equal to the real uid (gid). Turning this off causes the effective uid and gid to be set to the real uid and gid.

`-S` Prefixes commands with a date and time stamp of the form *day month date hh:mm:ss.*

`-s` Sorts the positional parameters lexicographically.
Note: For a decsription of `ksh -s`, see the Invocation subsection.

`-t` Exits after reading and executing one command.

`-u` Treats unset parameters as an error when substituting.

`-v` Prints shell input lines as they are read.

`-x` Prints commands and their arguments as they are executed.

`-` Turns off `-x`, `-v`, and `-S` flags and stops examining arguments for flags.

`--` Does not change any of the flags. This flag is useful in setting `$1` to a value beginning with `-`. If no arguments follow this flag, the positional parameters are unset.

Using `+` rather than `-` causes these flags to be turned off. These flags can also be used upon invocation of the shell. The current set of flags may be found in `$-`. Unless `-A` is specified, the remaining arguments are positional parameters and are assigned, in order, to `$1 $2` . . . . If no arguments are specified, the names and values of all named parameters are printed on the standard output.

`setucat` *cat* Sets the active category. For usage and description, see `setucat`(1).

`setusrv` Sets the user's security attributes. For usage and description, see `setusrv`(1).

`setucmp` *cmp* Sets active compartments. Available only to the lowest-level login shell. For usage and description, see `setucmp`(1).

`setulvl` *level* Raises the security level. Available only to the lowest-level login shell. For usage and description, see `setulvl`(1).

† `shift` [*n*] The positional parameters from `$`*n*`+1` ... are renamed `$1` ...; default *n* is 1. The parameter *n* can be any arithmetic expression that evaluates to a nonnegative number less than or equal to `$#`.

† `times` Prints the accumulated user and system times for the shell and for processes run from the shell.

† `trap` [*arg*] [*sig*] ...
arg is a command to be read and executed when the shell receives signal(s) *sig*. (Note that *arg* is scanned once when the trap is set and once when the trap is taken.) Each *sig* can be specified as a number or as the name of the signal. Trap commands are executed in order of signal number. Any attempt to set a trap on a signal that was ignored on entry to the current shell is ineffective. If *arg* is omitted or is `-`, all trap(s) *sig* are reset to their original values. If *arg* is the null string, this signal is ignored by the shell and by the commands it invokes. If *sig* is `ERR`, *arg* will be executed whenever a command has a nonzero exit status. If *sig* is `DEBUG`, *arg* will be executed after each command. If *sig* is `0` or `EXIT` and the `trap` statement is executed inside the body of a function, the command *arg* is executed after the function completes. If *sig* is `0` or `EXIT` for a `trap` set outside any function, the command *arg* is executed on exit from the shell. The `trap` command with no arguments prints a list of commands associated with each signal number.

†† `typeset` [±`HLRZfilrtux`[*n*]] [*name*[=*value*]] ...
Sets attributes and values for shell parameters. When invoked inside a function, a new instance of the parameter *name* is created. The parameter value and type are restored when the function completes. The following list of attributes may be specified:

`-H` This flag provides UNIX system to host-name file mapping on non-UNIX system machines.

`-L` Left justifies and removes leading blanks from *value*. If *n* is nonzero, it defines the width of the field; otherwise, it is determined by the width of the value of first assignment. When the parameter is assigned to, it is filled on the right with blanks or truncated, if necessary, to fit into the field. Leading 0's are removed if the `-Z` flag is also set. The `-R` flag is turned off.

`-R` Right justifies and fills with leading blanks. If *n* is nonzero, it defines the width of the field; otherwise, it is determined by the width of the value of first assignment. The field is left-filled with blanks or truncated from the end if the parameter is reassigned. The `L` flag is turned off.

`-Z` Right justifies and fills with leading 0's, if the first nonblank character is a digit and the `-L` flag has not been set. If *n* is nonzero, it defines the width of the field; otherwise, it is determined by the width of the value of first assignment.

`-f` The names refer to function names rather than parameter names. No assignments can be made and the only other valid flags are `-t`, `-u` and `-x`. The `-t` flag turns on execution tracing for this function. The `-u` flag causes this function to be marked undefined. The `FPATH` variable will be searched to find the function definition when the function is referenced. The `-x` flag allows the function definition to remain in effect across shell procedures invoked by name.

`-i` Specifies that parameter is an integer. This makes arithmetic faster. If *n* is nonzero, it defines the output arithmetic base; otherwise, the first assignment determines the output base.

`-l` Converts all uppercase characters to lowercase. The uppercase flag, `-u` is turned off.

`-r` The given *names* are marked read only; these names cannot be changed by subsequent assignment.

`-t` Tags the named parameters. Tags are user-definable and have no special meaning to the shell.

`-u` Converts all lowercase characters to uppercase characters. The lowercase flag, `-l`, is turned off.

`-x` Marks the given *name*s for automatic export to the environment of subsequently-executed commands.

Using `+` rather than `-` causes these flags to be turned off. If no *name* arguments are given but flags are specified, a list of names (and optionally the values) of the parameters that have these flags set is printed. (Using `+` rather than `-` keeps the values from being printed.) If no *name*s and flags are specified, the *names* and *attributes* of all parameters are printed.

`ulimit` [`-f`] [*n*] Imposes a size limit of *n* blocks. The `-f` option imposes a size limit of *n* 4096-byte blocks on files written by child processes (files of any size may be read). With no argument, the current limit in 4096-byte blocks is printed. See `ulimit`(2). If no option is specified, `-f` is assumed.

`umask` [*mask*] The user file-creation mask is set to *mask* (see `umask`(2)). *mask* can either be an octal number or a symbolic value as described in `chmod`(1). If a symbolic value is specified, the new umask value is the complement of the result of applying *mask* to the complement of the previous umask value. If *mask* is omitted, the current value of the mask is printed.

`unalias` *name* ...
The parameters given by the list of *name*s are removed from the *alias* list. See `unalias`(1).

`unset` [`-f`] *name* ...
The parameters given by the list of *name*s are unassigned; that is, their values and attributes are erased. Read-only variables cannot be unset. If the `-f` flag is set, the names refer to *function* names. Unsetting `ERRNO`, `LINENO`, `MAILCHECK`, `OPTARG`, `OPTIND`, `RANDOM`, `SECONDS`, `TMOUT`, and _ removes their special meaning, even if they are subsequently assigned to.

† `wait` [*job*] Waits for the specified *job* and reports its termination status. If *job* is not specified, all currently active child processes are awaited. The exit status from this command is that of the process awaited. For a format description of *job*, see the Jobs subsection. See `wait`(1).

`whence` [`-pv`] *name* ...
For each *name*, indicates how it would be interpreted if used as a command name.

The `-v` flag produces a more verbose report.

The `-p` flag does a path search for *name*, even if name is an alias, a function, or a reserved word.

NOTES

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
`secadm`, `system`	Can redirect to or from any file using the facilities described in the Parameter Substitution, Quoting, and Input/Output subsections of this man page; change to any directory using the `cd`(1) command described in the File Name Generation subsection of this man page; kill any user process using the `kill`(1) command; arbitrarily set the shell process security attributes using the `setulvl`(1), `setucmp`(1), and `setusrv`(1) commands; and change process limits to any value using the `ulimit`(2) command.
`sysadm`	Constrained by security label restrictions but not by ownership, mode, and ACL considerations when redirecting to or from files using the facilities described in the Parameter Substitution, Quoting, and Input/Output sections of this man page; can change to directories using `cd`(1) command; can expand path names using the patterns described in the File Name Generation subsection of this man page; or can kill user processes using the `kill`(1) command. The `sysadm` administrator can change process limits to any value using the `ulimit`(2) command. However, this user can only set the shell process security attributes using `setulvl`(1), `setucmp`(1), or `setusrv`(1) in ways that are allowed to nonadministrative users.

`sysops` Constrained by security label restrictions but not by ownership considerations when killing processes using the `kill`(1) command and can change process limits to any value using the `ulimit`(2) command. The `sysops` administrator is treated as a nonadministrative user with respect all other shell activities.

If the `PRIV_SU` configuration option is enabled, the super user can override all `ksh` restrictions.

If a command that is a tracked alias is executed, and a command with the same name is installed in a directory in the search path before the directory where the original command was found, the shell will continue to `exec` the original command. Use the `-t` option of the `alias`(1) command to correct this situation.

Some very old shell scripts contain a `^` as a synonym for the pipe character (`|`).

Using the `fc` built-in command within a compound command will cause the entire command to disappear from the history file.

The built-in command `.` *file* reads the entire file before any commands are executed. Therefore, `alias` and `unalias` commands in the file will not apply to any functions defined in the file.

Traps are not processed while a job is waiting for a foreground process. Thus, a trap on `CHLD` will not be executed until the foreground job terminates.

EXIT STATUS

Errors detected by the shell, such as syntax errors, cause the shell to return a nonzero exit status. If a specified shell script could not be found by a noninteractive shell, the shell exits with 127. Otherwise, the shell returns the exit status of the last command executed (see also the `exit` command). If the shell is being used noninteractively, execution of the shell file is abandoned. Run-time errors detected by the shell are reported by printing the command or function name and the error condition. If the line number that the error occurred on is greater than one, the line number is also printed in square brackets (`[ ]`) after the command or function name.

EXAMPLES

Example 1: The following is an example of setting up `.profile`, which is the initialization file for `ksh`. It is executed once at login, and then executes the file to which `ENV` is set.

```
CDPATH=.:..:~:/         #path that cd uses
FCEDIT=vi               #command line editor
ENV=~/.env              #environment file
HISTSIZE=32             #ksh saves last 32 commands
MAILCHECK=300           #ksh checks for mail every 5 minutes
set -o ignoreeof        #ignore ^D on login shell
TERM='tset - -Q '?ampex230''
stty erase '^?' kill '^u' intr '^c' echo

export CDPATH EDITOR FCEDIT CRAY ENV HISTSIZE MAILCHECK TERM
```

Example 2: The following is an example of a `.env` file that is executed each time a new `ksh` is run; it allows all aliases to be carried to all new shells.

```
#
alias -x lsf='/bin/ls -CF'
alias -x lsl='/bin/ls -lgF'
alias -x h='fc -lr'
alias -x hf='fc -l $HISTSIZE | more'
alias -x pe=printenv
```

FILES

`/etc/profile`	File containing system default shell startup commands
`$HOME/.profile`	File containing user's shell startup commands
`$TMPDIR/sh*`	Temporary working files
`/dev/null`	Zero-length file
`a.out`	Executable binary file

SEE ALSO

`alias`(1), `bg`(1), `cd`(1), `chown`(1), `command`(1), `echo`(1), `emacs`(1), `env`(1), `fc`(1), `fg`(1), `getopts`(1), `jobs`(1), `kill`(1), `login`(1), `pwd`(1), `read`(1), `setucat`(1), `setucmp`(1), `setusrv`(1), `test`(1), `unalias`(1), `vi`(1), `wait`(1)

`dup`(2), `exec`(2), `execve`(2), `fork`(2), `pipe`(2), `setuid`(2), `signal`(2), `umask`(2), `ulimit`(2), `wait`(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

`a.out`(5), `profile`(5) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

`dmmode`(1) Online only

`environ`(7) Online only

Learning the Korn Shell, Bill Rosenblatt, O'Reilly & Associates, Inc., 1990

The KornShell Command and Programming Language, Morris Bolsky and David Korn, Prentice Hall, 1989

General UNICOS System Administration, Cray Research publication SG–2301

NAME

ksrvtgt – Uses a service key to fetch and store a Kerberos ticket-granting ticket

SYNOPSIS

ksrvtgt*name instance* [[*realm*] *srvtab*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The ksrvtgt utility retrieves a ticket-granting ticket with a lifetime of 5 minutes for the principal *name.instance@realm*. It uses the service key found in the *srvtab* file to decrypt the response and stores the ticket in the standard ticket cache.

The ksrvtgt utility accepts the following options:

name instance
Specifies the principal for which the ticket-granting ticket is retrieved.

realm Specifies the realm of the principal for which the ticket-granting ticket is retrieved. If you omit *realm* on the command line, the local realm is used.

srvtab Specifies the file that contains the service key that decrypts the response. If you omit *srvtab* on the command line, the /etc/srvtab file is used.

The ksrvtgt utility is primarily for use in shell scripts and other batch-type facilities.

MESSAGES

The Generic kerberos failure (kfailure) message can indicate a whole range of problems, the most common of which is the inability to read the service key file.

FILES

/etc/krb.conf	File that gets the name of the local realm
/etc/srvtab	Default service key file
/tmp/tkt[*uid*]	Default ticket file

SEE ALSO

kdestroy(1), kinit(1)

kerberos(7) (available only online)

NAME

ksu – Uses Kerberos to substitute user ID

SYNOPSIS

ksu [-] [*name* [*args*]]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The ksu utility allows you to become another user without logging off. The default user *name* is root.

To use ksu, the appropriate password must be supplied (unless you are an appropriately authorized user). If the password is correct, ksu executes a new shell with the real and effective user ID set to that of the specified user. The new shell is the optional program named in the shell field of the specified user's password file entry (see udb(5)), or /bin/sh if none is specified (see sh(1)). To restore your original user identity, exit the new shell.

Any additional arguments specified on the command line are passed to the program invoked as the shell. For example, when sh(1) is used, an argument of the form -c *string* executes *string* via the shell and an option of -r gives the user a restricted shell.

The following statements are true only if the optional program named in the shell field of the specified user's password file entry is like sh(1). If the first argument to ksu is a -, the environment is changed to what would be expected if the user actually logged in as the specified user. This is done by invoking the program used as the shell with an *arg0* value whose first character is -, causing the system's profile (/etc/profile) and then the specified user's profile (.profile in the new home directory) to be executed. Otherwise, the environment is passed along, with the possible exception of $PATH, which is set to /bin:/etc:/usr/bin for root. If the optional program used as the shell is /bin/sh, the user's .profile can check *arg0* for -sh or -ksu to determine whether it was invoked by login(1) or ksu, respectively. If the user's program is not /bin/sh, the program is invoked with an *arg0* of -*program* by both login and ksu.

The ksu utility accepts the following options:

- Changes environment to that of specified user name.

name Indicates user name to log on to (default is root).

args Specifies shell arguments for new login.

Only users with root instances listed in the `.klogin` file can use the `ksu` utility to change to `root` (the `klogin`(1) utility describes the format of this file). When you attempt root access, `ksu` attempts to fetch a ticket-granting ticket for *username.root@localrealm*; *username* is the user name of the process. If possible, the tickets are used to obtain, use, and verify tickets for the `rcmd.host@`*localrealm* service; `host` is the canonical host name of the machine (which is the first field, lower case, of the domain name). If this verification fails, the `ksu` utility is disallowed. If the `rcmd.host@`*localrealm* service is not registered, the `ksu` utility is allowed.

By default (unless the prompt is reset by a startup file), the super-user prompt is set to #.

When not attempting to switch to the `root` user, `ksu` behaves exactly like `su`(1).

SEE ALSO

`csh`(1), `klogin`(1), `login`(1), `passwd`(1), `sh`(1), `su`(1)

`group`(5) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

`environ`(7) (available only online)

NAME

`last` – Indicates the last logins of users and teletypes

SYNOPSIS

`/usr/ucb/last` [`-`*number*] [`-f` *file*] [*names*] [*ttys*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `last` utility looks in the `/etc/wtmp` file for information about a user, a teletype, or any group of users and teletypes. The `wtmp` file records all logins and logouts that have occurred since the last initialization of the file. Arguments specify names of users or teletypes of interest. Names of teletypes can be specified fully or abbreviated. For example, `last 0` is the same as `last tty0`. If you specify multiple arguments, `last` prints the information applying to any of the arguments. The `last` utility prints the sessions of the specified users and teletypes, most recent first, indicating the times at which the session began, the duration of the session, and the teletype on which the session occurred. `last` indicates whether the session is still continuing or was cut short by a reboot.

The pseudo-user `reboot` logs in when the system is rebooted. Thus, the following command line indicates the mean time between reboots:

```
last reboot
```

The `last` utility accepts the following options:

`-`*number* Limits the number of entries displayed to *number*.

`-f` *file* Uses *file* as the name of the accounting file instead of `/etc/wtmp`. *file* must be in `wtmp`(5) format.

names Logins to be checked.

ttys Teletypes to be checked.

EXAMPLES

Example 1: To list all of the super user's sessions, as well as all sessions on the console terminal, enter the following:

```
last root console
```

Example 2: To print a record of all logins and logouts in reverse order, enter the following without arguments:

```
last
```

FILES

`/etc/wtmp` Login database

SEE ALSO

`utmp`(5) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

NAME

ld – Invokes the link editor

SYNOPSIS

ld [-D *dirstring*] [-e *name*] [-F] [-g] [-i] [-j *names*] [-l *names*] [-L *ldirs*] [-m] [-n] [-o *outfile*] [-r] [-s] [-u *unames*] [-V] [-Z] [-z *file*] *files*

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The ld command links relocatable object modules to produce an executable program. This command invokes the same loader as does segldr(1), but with a traditional UNIX ld invocation.

The *files* specified on the command line may be either sequential object files created by the compilers or assembler, object libraries created by bld(1) or files containing loader directives. Files ending with .o will be treated as bin files. Files ending with .a will be treated as lib files. You can intersperse file names with options on the command line.

The ld command accepts the following options:

-D *dirstring* Specifies a character string composed of loader directives separated by semicolons. The loader processes directives supplied with -D before it processes any directives files.

-e *name* Sets the program entry address to the value of the symbol *name*.

-F Enables default library processing. The standard system libraries are processed after any user-supplied libraries. Processing of the system libraries is disabled by default.

-g Generates the Debug Symbol tables and appends them to the executable program. This option is enabled by default. See the -s option.

-i Generates a shared-text program on Cray PVP systems. This option is equivalent to the -n option.

-j *names* List of directives file names, separated by commas. When a name begins with a dot (.) or a slash (/), ld assumes it is a complete path name and uses it without modification. Otherwise, ld checks for a segdirs/*name* file in the list of search directories and uses the first one found. See the -L option for the list of search directories.

-l *names* Identifies library files. When a name begins with a dot (.) or a slash (/), it is assumed be a complete path name and is used without modification. Otherwise, the ld command checks first for files /opt/ctl/craylibs/craylibs/lib*name*.a and /lib/lib*name*.a, and then for file /usr/lib/lib*name*.a. It uses the first file it finds. See the -L option.

`-L` *ldirs* Changes the `-l` option search algorithm to look for library files in directories *ldirs* before looking in the `/opt/ctl/craylibs/craylibs`, `/lib`, or `/usr/lib` directories. If the `-F` option is used to include the system default directories, the loader searches directories *ldirs* for those libraries before searching the `/opt/ctl/craylibs/craylibs`, `/lib`, or `/usr/lib` directories. Multiple `-L` options are cumulative.

`-m` Generates a load map of the executable program and writes it to the `stdout` file.

`-n` Generates a shared-text program on Cray PVP systems. This option is equivalent to the `-i` option.

`-o` *outfile* Writes the executable program to *outfile*. The default *outfile* name is `a.out`.

`-r` Produces a relocatable output from `.o` files. That is, instead of generating an executable, it generates one relocatable combining the .o files named. The output is suitable for use by another invocation of `ld`. Equivalent to using the following directives:

```
OUTFORM=REL
USX=NOTE
SYSTEM=STDALONE
ZSYMS=OFF
```

`-s` Inhibits the generation of the Debug Symbol tables.

`-u` *unames* Enters *unames* as undefined symbols. This is useful for loading from a library, because undefined symbols are needed to force the loading of desired routines.

`-V` Lists the SEGLDR version line on the `stderr` file.

`-Z` Inhibits the loader from reading the default directives file. The default directives file is either `/opt/ctl/craylibs/craylibs/segdirs/opt_defld` or `/lib/segdirs/def_ld`. The default directives file is required for configuring programs correctly for execution under the UNICOS operating system. The `-Z` option should be used only by special-purpose programs.

`-z` *file* Specifies an alternative default directives file. The alternative directives must configure the program correctly for execution under the UNICOS operating system.

files Specifies the files to be loaded.

ENVIRONMENT VARIABLES

The `ld` command looks for and processes the following environment variables:

`LDDIR` Contains one or more strings separated by semicolons. Each string may be either a `ld` directive or the name of a file containing `ld` directives.

`TMPDIR` Specifies the directory that the loader uses for its temporary file.

`LPP` Specifies the number of lines to print on each page of listing output. The value must be between 15 and 999, and the default is 57.

MSG_FORMAT Describes a format specification similar to that of the C library routine `printf`; this specification can be used to alter `ld` error message displays.

NLSPATH Specifies a list of alternate directories that the loader should search for its error message catalog. It is used to select alternate catalogs for debugging, or when different versions of `ld` are operating on the same system. `NLSPATH` is not needed for normal operations.

TARGET Specifies the machine characteristics of the system on which the program will execute. If the `TARGET` variable has not been specified, the program will be adapted to the host system.

The following defaults for loader directives are automatically used when you invoke `ld`:

```
force=on
duporder=on
nodeflib
dupentry=caution:note:note
usx=warning
```

MESSAGES

The full range of error messages and the proper responses are listed in the *Segment Loader (SEGLDR) and ld Reference Manual*, Cray Research publication SR–0066.

FILES

`a.out`	Executable program
file`.o`	Relocatable object file
`/opt/ctl/craylibs/craylibs/libf.a`	Fortran library
`/opt/ctl/craylibs/craylibs/libfi.a`	Fortran intrinsic library
`/opt/ctl/craylibs/craylibs/libm.a`	Math library
`/opt/ctl/craylibs/craylibs/libsci.a`	Scientific library
`/lib/libc.a`	C library
`/opt/ctl/CC/CC/lib/libC.a`	C++ library (only if your site has a C++ license)
`/lib/libp.a`	Pascal library
`/opt/ctl/craylibs/craylibs/libu.a`	Utility library
`/lib/segdirs/def_ld` and `/opt/ctl/craylibs/craylibs/segdirs/opt_defld`	Default directives files

SEE ALSO

`ar`(1) archive and library maintainer for portable archives
`bld`(1) maintains relocatable libraries
`nm`(1) prints name list from load modules
`segldr`(1) invokes the Cray Research segment loader (SEGLDR)

`cc`(1) invokes the Cray Standard C compiler and is described in the *Cray Standard C Reference Manual*, Cray Research publication SR–2074

`f90`(1) invokes the CF90 compiler and is described in the *CF90 Commands and Directives Reference Manual*, Cray Research publication SR–3901

`a.out`(5) describes the loader output file
`relo`(5) describes the relocatable object table format under the UNICOS operating system
`taskcom`(5) describes the task common table format
in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

Segment Loader (SEGLDR) and ld Reference Manual, Cray Research publication SR–0066

NAME

`lex` – Generates programs for simple lexical tasks

SYNOPSIS

```
lex [-t] [-r] [-n] [files]
lex [-t] [-r] [-v] [files]
```

Obsolescent version; may not be supported in future releases:

```
lex -c [-t] [-r] [-n] [files]
lex -c [-t] [-r] [-v] [files]
```

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4
AT&T extensions (`-r` option)

DESCRIPTION

The `lex` utility generates programs to be used in simple lexical analysis of text.

The input files (standard input by default) contain strings and expressions for which to search and C text to be executed when strings are found.

The `lex` utility accepts the following options and operand:

`-t` Causes the `lex.yy.c` program to be written to standard output.

`-r` Indicates RATFOR actions (RATFOR, a Rational Fortran compiler, is not supported by Cray Research.)

`-n` Does not print the `-v` summary.

`-v` Provides a one-line summary of statistics of the machine generated. Multiple files are treated as a single file. When files are not specified, standard input is used.

`-c` (Obsolescent) Indicates C-language actions and is the default.

files A path name of one or more input files. If more than one such file is specified, all files are concatenated to produce a single `lex` program. If no *files* are specified, or if the operand is `-`, the standard input is used.

A file, `lex.yy.c`, is generated; when loaded with the library, it copies the input to the output except when a string specified in the file is found. Then the corresponding program text is executed. The actual string matched is left in `yytext`, an external character array. Matching is done in order of the strings in the file. The strings may contain brackets to indicate character classes, as in `[abx-z]` to indicate `a`, `b`, `x`, `y`, and `z`. The `*`, `+`, and `?` operators mean, respectively, any nonnegative number of, any positive number of, and either zero or one occurrence of, the previous character or character class. The `.` character is the class of all ASCII characters except newline characters.

Parentheses for grouping and the vertical bar for alternation are also supported. The notation *r* {`d`, `e`} in a rule indicates between `d` and `e` instances of extended regular expression *r* . It has higher precedence than `|`, but lower than `(`, `**`, `?` , `+`, and concatenation. Thus, `[a-zA-Z]+` matches a string of letters. The `^` character at the beginning of an expression permits a successful match only immediately after a newline character, and `$` at the end of an expression requires a trailing newline character.

The `/` character in an expression indicates trailing context; only the part of the expression up to the slash is returned in `yytext`, but the remainder of the expression must follow in the input stream. An operator character may be used as an ordinary symbol when it is enclosed by `"` symbols or preceded by `\`.

Three subroutines defined as macros are expected: `input()` to read a character; `unput`(*c*) to replace a character read; and `output`(*c*) to place an output character. They are defined in terms of the standard streams, but you can override them. The program generated is named `yylex`(), and the library contains a `main()` routine that calls it. The `REJECT` action on the right side of the rule causes this match to be rejected and the next suitable match executed; the `yymore()` function accumulates additional characters into the same `yytext` array; and the `yyless(p)` function pushes back the portion of the string matched beginning at *p*, which should be between `yytext` and `yytext` + `yyleng`. The `input` and `output` macros, defaulted to `stdin` and `stdout`, respectively, use files `yyin` and `yyout` to read from and write to, respectively.

Any line beginning with a `<blank>` is assumed to contain only C text and is copied; when it precedes `%%`, it is copied into the external definition area of the `lex.yy.c` file. All rules should follow a `%%` (as is done in `yacc`(1)). Lines that precede `%%` and begin other than a `<blank>` character define the string on the left as the remainder of the line; the string can be called out later if it is surrounded with `{}`. Braces do not imply parentheses; only string substitution is done.

Certain table sizes for the resulting finite-state machine can be set in the definitions section as follows:

`%p` *n* Number of positions is *n* (default is 2000).

`%n` *n* Number of states is *n* (default is 500).

`%t` *n* Number of parse tree nodes is *n* (default is 1000).

`%a` *n* Number of transitions is *n* (default is 3000).

The use of one or more of these automatically implies the `-v` option, unless the `-n` option is used.

NOTES

The external names generated by `lex` all begin with the prefix `yy` or `YY`.

EXIT STATUS

The `lex` utility exits with one of the following values:

0 Successful completion.

>0 An error occurred.

BUGS

Extended Regular Expressions (EREs) are currently not supported in `lex`.

EXAMPLES

Example 1: The following example is an input program for `lex`:

```
D    [0-9]
%%
if      printf("IF statement0);
[a-z]+  printf("tag, value %s0,yytext);
0{D}+   printf("octal number %s0,yytext);
{D}+    printf("decimal number %s0,yytext);
"++"    printf("unary op0);
"+"     printf("binary op0);
"/*" skipcommnts();
%%
  skipcommnts()
 {
        for (;;)
        {
                while (input() != '*')
                        ;
         if (input() != '/')
                        unput(yytext[yyleng-1]);
                else
                        return;
         }
 }
```

SEE ALSO

yacc(1)

lex & yacc, Doug Brown and Tony Mason, O'Reilly & Associates, Inc., 1992.

The UNIX Programming Environment, Brian W. Kernighan and Rob Pike, Prentice-Hall, Inc., 1984.

NAME

`limit` – Sets resource limits

SYNOPSIS

`limit` [`-j` *jid*] [`-c` *cputimelim*] [`-m` *memorylim*] [`-e` *mpppelim*] [`-t` *mpptimelim*] [`-s` *socketbuflim*] [`-v`]

`limit -p` *pid* [`-c` *cputimelim*] [`-m` *memorylim*] [`-d` *corelim*] [`-f` *openfilelim*] [`-t` *mpptimelim*] [`-s` *socketbuflim*] [`-v`]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `limit` utility establishes limits on resource usage for a process or a job.

The `limit` utility accepts the following options:

`-c` *cputimelim* Indicates a limit on CPU time. The *cputimelim* argument refers to CPU seconds. A *cputimelim* of 0 indicates an unlimited amount of CPU time. If the `-c` option is not specified, the CPU time limit is not modified. (See the CAUTIONS section.)

`-d` *corelim* Indicates a limit on core file sizes. The *corelim* argument refers to memory words and is rounded up to the nearest click boundary. There are 512 decimal words per click on Cray Research systems. A *corelim* of 0 indicates the maximum core file size allowed. Specifying a *corelim* value less than the size of the process will result in a truncated core file that consists only of the user common structure and the user area. Specifying a *corelim* of `nocore` will inhibit the creation of a core file altogether. This option is supported only for processes.

`-e` *mpppelim* Indicates a limit on the number of PEs. This option may be specified only with the `-j` option.

`-f` *openfilelim* Indicates a limit on the maximum number of files that a process can have open at any given time. This limit is supported only with the `-p` option. The limit may be from 64 through the user's defined limit in the user database (UDB); the default is 64. This limit is applied only to the children of the process specified.

`-j` *jid* Indicates a particular job. A *jid* of 0 means the current job. The `-j` option may not be specified with the `-p` option. If neither the `-p` nor the `-j` option is specified, a default of `-j 0` is used.

`-m` *memorylim* Indicates a limit on memory size. The *memorylim* argument refers to memory words. The given *memorylim* is rounded up to the nearest click boundary. There are 512 decimal words per click on Cray Research systems. A *memorylim* of 0 indicates the maximum memory size available. If the `-m` option is not specified, the memory size limit is not modified.

`-p` *pid* Indicates a particular process. A *pid* of 0 means the current process. The `-p` option may not be specified with the `-j` option.

`-s` *socketbuflim* Indicates a limit on per session socket buffer (sockbuf) space. The per session sockbuf space is the sum of the sockbuf space requested by all of the sockets used by the session. The *socketbuflim* argument refers to memory clicks. There are 512 decimal words per click on Cray Research systems. A *socketbuflim* of 0 indicates no space limit. If the `-s` option is not specified, the sockbuf space limit is not modified.

`-t` *mpptimelim* Indicates a limit on processing element (PE) time.

`-v` Writes the previous time, memory size, and core file size limits to standard output in a more verbose mode.

Any user may change a limit to be more restrictive. Only an appropriately authorized user can increase resource limits. Only an appropriately authorized user can set the resource limits of another user, process, or session. Limits are inherited by child processes.

The *pid*, *jid*, *cputimelim*, *corelim*, *memorylim*, and *socketbuflim* arguments have the following characteristics in common. All must be nonnegative integer values. If the argument contains a leading `0x` or `0X`, it is evaluated as hexadecimal. If the argument contains a leading zero, it is evaluated as octal. Otherwise, it is evaluated as decimal.

NOTES

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
`system`, `secadm`	Allowed to raise or lower the resource limits of any user, process, or session.
`sysadm`	Allowed to raise or lower the resource limits of any user, process, or session, subject to security label restrictions. Shell-redirected I/O is subject to security label restrictions.

If the `PRIV_SU` configuration option is enabled, the super user is allowed to raise or lower the resource limits of any user, process, or session.

If a process exceeds the process CPU time limit, `SIGCPULIM` is sent to the offending process. By default, the `SIGCPULIM` will terminate the process and the parent shell will recognize the `SIGCPULIM` and send an error message to standard error. If the job CPU time limit is exceeded, `SIGCPULIM` is sent to all processes in the job, which includes the parent shell. In this case, no error message will be sent to standard error.

CAUTIONS

If more than one call is made to `limit` at nearly the same time to modify the same entity, there is potential for unpredictable results. The CPU time limit does not apply when running as root.

EXIT STATUS

If `limit` succeeds, several decimal integers (separated by a newline character) are written to `stdout`. If core files are disabled, the string `nocore` is printed.

When the job mode (`-j`) option is specified, the following values are written to `stdout`:

- CPU seconds
- Words of memory
- Socket buffer limit
- Number of PE limit
- PE time limit

When the process mode (`-p`) option is specified, the following values are written to `stdout`:

- CPU seconds
- Words of memory
- Core file limit
- File descriptor limit
- Socket buffer limit
- PE time limit

The `limit` utility returns an exit status of 0 upon success.

If `limit` fails, an appropriate error message is written to `stderr`, and a nonzero exit status is returned. When `limit` fails, none of the limits are modified, if possible.

SEE ALSO

`nlimit`(1)

`limit`(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

NAME

`line` – Reads one line

SYNOPSIS

```
line
```

IMPLEMENTATION

All Cray Research systems

STANDARDS

XPG4

DESCRIPTION

The `line` utility copies one line (up to a `<newline>` character) from the standard input and writes it on standard output.

It returns an exit code of 1 on `EOF` and always prints at least a `<newline>` character. It is often used in shell scripts to read from the user's terminal.

NOTES

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
`system`, `secadm`	In a privileged administrator shell environment, shell-redirected I/O is not subject to file protections.

If the `PRIV_SU` configuration option is enabled, shell-redirected I/O on behalf of the super user is not subject to file protections.

The `read`(1) utility is the preferred method of obtaining input from a user's terminal.

EXIT STATUS

The `line` utility exits with one of the following values:

0 Successful completion.

>0 An error occurred.

EXAMPLES

The following standard shell (sh(1)) script reads input lines from stdin and writes them to file ofile:

```
echo "Enter text.  End with CONTROL-d"
while REC="`line`"
do
        echo "${REC}" >> ofile
done
```

SEE ALSO

read(1), sh(1)

read(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

NAME

`lint` – A C-language program checker

SYNOPSIS

`lint` [`-a`] [`-b`] [`-c`] [`-D` *name* = *value*] [`-F`] [`-h`] [`-I` *directory*] [`-k`] [`-L` *directory*] [`-m`] [`-n`] [`-o` *x*] [`-p`] [`-s`] [`-u`] [`-U` *name*] [`-x`] [`-v`] [`-V`] [`-y`] *files*

IMPLEMENTATION

All Cray Research systems

STANDARDS

XPG4 AT&T extensions (`-F`, `-h`, `-k`, `-m`, `-s`, `-V`, and `-y` options)

DESCRIPTION

`lint` detects features of C program files that are likely to be bugs, nonportable, or wasteful. It also checks type usage more strictly than the compiler. `lint` issues error and warning messages. Among the things it detects are unreachable statements, loops not entered at the top, automatic variables declared and not used, and logical expressions whose value is constant. `lint` checks for functions that return values in some places and not in others, functions called with varying numbers or types of arguments, and functions whose values are not used or whose values are used but none returned.

Arguments whose names end with `.c` are taken to be C source files. Arguments whose names end with `.ln` are taken to be the result of an earlier invocation of `lint`, with either the `-c` or the `-o` option used. The `.ln` files are analogous to `.o` (object) files that are produced by the `cc`(1) utility when given a `.c` file as input. Files with other suffixes are warned about and ignored.

`lint` takes all the `.c`, `.ln`, and `llib-l`*x*`.ln` (specified by `-l`*x*) files and processes them in their command-line order. By default, `lint` appends the standard C `lint` library (`llib-lc.ln`) to the end of the list of files. When the `-c` option is used, the `.ln` and the `llib-l`*x*`.ln` files are ignored. When the `-c` option is not used, the second pass of `lint` checks the `.ln` *and the* `llib-l`*x*`.ln` list of files for mutual compatibility.

Any number of `lint` options may be used, in any order, intermixed with file name arguments. The following options are supported:

`-a` Suppresses complaints about assignments of long values to variables that are not long.

`-b` Suppresses complaints about `break` statements that cannot be reached.

`-h` Does not apply heuristic tests that attempt to intuit bugs, improve style, and reduce waste.

`-m` Suppresses complaints about external symbols that could be declared static.

`-u` Suppresses complaints about functions and external variables used and not defined, or defined and not used. (This option is suitable for running `lint` on a subset of files of a larger program.)

`-v` Suppress complaints about unused arguments in functions.

`-x` Does not report variables referred to by external declarations but never used.

The following arguments alter `lint`'s behavior:

`-I` *directory*
Searches for included header files in the directory *directory* before searching the current directory and/or the standard place.

`-l`*x* Includes the `lint` library `llib-l`*x*`.ln`. For example, you can include a `lint` version of the math library `llib-lm.ln` by inserting `-lm` on the command line. This argument does not suppress the default use of `llib-lc.ln`. These `lint` libraries must be in the assumed directory. This option can be used to reference local `lint` libraries and is useful in the development of multi-file projects.

`-L` *directory*
Searches for `lint` libraries in *directory* before searching the standard place.

`-n` Does not check compatibility against the standard C `lint` library.

`-p` Attempts to check portability to other dialects of C. Along with stricter checking, this option causes all nonexternal names to be truncated to 8 characters and all external names to be truncated to 6 characters and one case.

`-s` Produces one-line diagnostics only. `lint` occasionally buffers messages to produce a compound report.

`-k` Alters the behavior of `/*LINTED [`*message*`]*/` directives. Normally, `lint` will suppress warning messages for the code following these directives. Instead of suppressing the messages, `lint` prints an additional message containing the comment inside the directive.

`-y` Specifies that the file being checked by `lint` will be treated as if the `/*LINTLIBRARY*/` directive had been used. A `lint` library is normally created by using the `/*LINTLIBRARY*/` directive.

`-F` Prints path names of files. `lint` normally prints the file name without the path.

`-c` Causes `lint` to produce a `.ln` file for every `.c` file on the command line. These `.ln` files are the product of `lint`'s first pass only, and are not checked for interfunction compatibility.

`-o` *x* Causes `lint` to create a `lint` library with the name `llib-l`*x*`.ln`. The `-c` option nullifies any use of the `-o` option. The `lint` library produced is the input that is given to `lint`'s second pass. The `-o` option simply causes this file to be saved in the named `lint` library. To produce a `llib-l`*x*`.ln` without extraneous messages, use of the `-x` option is suggested. The `-v` option is useful if the source file(s) for the `lint` library are just external interfaces.

Some of the above settings are also available through the use of "`lint` comments" (see below).

`-V` Writes to standard error the product name and release.

The following options are not for general use:

`-R` *file* Writes a `.ln` file to *file*, for use by `cxref`(1).

`-W` *file* Writes a `.ln` file to *file*, for use by `cflow`(1).

`lint` recognizes many `cc`(1) command-line options, including `-D`, `-U`, `-g`, and `-O`, although `-g` and `-O` are ignored. Unrecognized options are warned about and ignored. The predefined macro `lint` is defined to allow certain questionable code to be altered or removed for `lint`. Thus, the symbol `lint` should be thought of as a reserved word for all code that is planned to be checked by `lint`.

Certain conventional comments in the C source will change the behavior of `lint`:

`/*ARGSUSED`*n*`*/`	Makes `lint` check only the first `n` arguments for usage; a missing *n* is taken to be 0 (this option acts like the `-v` option for the next function).
`/*CONSTCOND*/` or `/*CONSTANTCOND*/` or `/*CONSTANTCONDITION*/`	Suppresses complaints about constant operands for the next expression.
`/*EMPTY*/`	Suppresses complaints about a null statement consequent on an if statement. This directive should be placed after the test expression, and before the semicolon. This directive is supplied to support empty if statements when a valid else statement follows. It suppresses messages on an empty `else` consequent.
`/*FALLTHRU*/` or `/*FALLTHROUGH*/`	Suppresses complaints about fall through to a `case` or `default` labeled statement. This directive should be placed immediately preceding the label.
`/*LINTLIBRARY*/`	At the beginning of a file, shuts off complaints about unused functions and function arguments in this file. This is equivalent to using the `-v` and `-x` options.
`/*LINTED` [*message*]`*/`	Suppresses any intrafile warning except those dealing with unused variables or functions. This directive should be placed on the line immediately preceding where the `lint` warning occurred. The `-k` option alters the way in which `lint` handles this directive. Instead of suppressing messages, `lint` will print an additional message, if any, contained in the comment. This directive is useful in conjunction with the `-s` option for post-`lint` filtering.
`/*NOTREACHED*/`	At appropriate points, stops comments about unreachable code. (This comment is typically placed just after calls to functions like `exit`(2)).
`/*PRINTFLIKE`*n*`*/`	Makes `lint` check the first (*n*–1) arguments as usual. The *n*th argument is interpreted as a `printf` format string that is used to check the remaining arguments.

/*PROTOLIB*n*/	Causes lint to treat function declaration prototypes as function definitions if *n* is nonzero. This directive can only be used in conjunction with the LINTLIBRARY directive. If *n* is zero, function prototypes will be treated normally.
/*SCANFLIKE*n*/	Makes lint check the first (*n*–1) arguments as usual. The *n*th argument is interpreted as a scanf format string that is used to check the remaining arguments.
/*VARARGS*n*/	Suppresses the usual checking for variable numbers of arguments in the following function declaration. The data types of the first *n* arguments are checked; a missing *n* is taken to be 0. The use of the ellipsis terminator (...) in the definition is suggested in new or updated code.

lint produces its first output on a per-source-file basis. Complaints regarding included files are collected and printed after all source files have been processed, if -s is not specified. Finally, if the -c option is not used, information gathered from all input files is collected and checked for consistency. At this point, if it is not clear whether a complaint stems from a given source file or from one of its included files, the source file name will be printed followed by a question mark.

The behavior of the -c and the -o options allows for incremental use of lint on a set of C source files. Generally, one invokes lint once for each source file with the -c option. Each of these invocations produces a .ln file that corresponds to the .c file, and prints all messages that are about just that source file. After all the source files have been separately run through lint, it is invoked once more (without the -c option), listing all the .ln files with the needed -l*x* options. This will print all the interfile inconsistencies. This scheme works well with make(1); it allows make(1) to be used to lint only the source files that have been modified since the last time the set of source files were checked by lint.

FILES

LIBDIR	The directory where the lint libraries specified by the -l*x* option must exist (default is /usr/lib/lint)
LIBDIR/lint[12]	First and second passes
LIBDIR/llib-lc.ln	Declarations for C library functions (binary format; source is in LIBDIR/llib-lc)
LIBPATH/llib-lm.ln	Declarations for math library functions (binary format; source is in LIBDIR/llib-lm)
TMPDIR/**lint*	Temporary files

SEE ALSO

cc(1), cflow(1), cxref(1), make(1)

exit(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

NAME

lmcksum – Checksums the FLEXlm license file

SYNOPSIS

lmcksum [-c *license_file*] [-k]

IMPLEMENTATION

All supported platforms

DESCRIPTION

The lmcksum command computes the checksum of the portions of the flexible license manager (FLEXlm) license file that end users cannot change. lmcksum computes a checksum for each line in the license file and an overall checksum of the license file. The following fields are used when creating the checksum:

- *hostid* on the server lines
- *daemon_name* on the daemon lines
- *feature_name*, *version*, *daemon_name*, *expiration_date*, *number_of_licenses*, *encryption_code*, *vendor_string*, and *hostid* on feature lines

The lmcksum command is available only in the FLEXlm v2.4 release and later. lmcksum is either a link to or a copy of the lmutil(1) utility.

The lmcksum command accepts the following options:

-c *license_file*
: Uses the specified *license_file* as input. If you omit the -c option, lmcksum uses the LM_LICENSE_FILE environment variable to find the license file to use. If that environment variable is not set, lmcksum uses the /usr/local/flexlm/licenses/license.dat file. If you omit the *license_file* argument, lmcksum uses the LM_LICENSE_FILE environment variable. If that environment variable is not set, lmcksum uses the /usr/local/flexlm/licenses/license.dat file.

-k
: Case-sensitive checksum; computes the checksum by using the exact case of the feature line(s) encryption codes.

SEE ALSO

lmutil(1) for information about the core FLEXlm utility

license.dat(5) for information about the license configuration file for FLEXlm licensed applications in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

NAME

lmdown – Shuts down all FLEXlm license daemons gracefully

SYNOPSIS

lmdown [-c *license_file*] [-q]

IMPLEMENTATION

All supported platforms

DESCRIPTION

The lmdown command sends a message to all flexible license manager (FLEXlm) license daemons asking them to shut down. The license daemons write out their last messages to the log file, close the file, and exit. All licenses that the license daemons have provided are rescinded, so that the next time a client program verifies the license, it will not be valid. In the FLEXlm v2.4 release and later, lmdown is either a link to or a copy of the lmutil(1) utility.

The lmdown command accepts the following options:

-c *license_file*
: Uses the specified *license_file* as input. On UNICOS systems, to avoid affecting other license managers, always use the -c option or the LM_LICENSE_FILE environment variable. If you omit the -c option, lmdown uses the LM_LICENSE_FILE environment variable to find the license file to use. If that environment variable is not set, lmdown uses the /usr/local/flexlm/licenses/license.dat file. If you omit the *license_file* argument, lmdown uses the LM_LICENSE_FILE environment variable. If that environment variable is not set, lmdown uses the /usr/local/flexlm/licenses/license.dat file.

-q
: (Quiet mode) If you specify this option, lmdown will not ask for confirmation before asking the license daemons to shut down. If you omit this option, lmdown asks for confirmation before asking the license daemons to shut down.

SEE ALSO

lmgrd(1) for information about invoking the FLEXlm daemon
lmreread(1) for information about instructing the FLEXlm license daemon to reread the license file
lmstat(1) for information about reporting status of FLEXlm license daemons and feature usage
lmutil(1) for information about the core FLEXlm utility

license.dat(5) for information about the license configuration file for FLEXlm licensed applications in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

NAME

lmgrd – Invokes the FLEXlm license daemon

SYNOPSIS

lmgrd [-2] [-b] [-c *license_file*] [-d] [-l *logfile*] [-p] [-s *interval*] [-t *timeout*]

IMPLEMENTATION

All supported platforms

DESCRIPTION

The lmgrd command invokes the flexible license manager (FLEXlm) license daemon and is the main daemon program for the FLEXlm distributed license management system. When you invoke lmgrd, it looks for a license file that contains all required information about vendors and features.

The lmgrd command accepts the following options:

-2 Specifies startup arguments; if you use the -p option, the -2 option is required. The -2 option is the opposite of the -b option. Available in lmgrd v2.4 and later.

-b Specifies backward-compatibility mode. In FLEXlm v2.4 or later, the -b option is the default. If you are running a v2.1 or later version of lmgrd with a v1.5 or earlier vendor daemon, use this option.

-c *license_file* Uses the specified *license_file* as input. If you omit the -c option, lmgrd uses the LM_LICENSE_FILE environment variable to find the license file to use. If that environment variable is not set, lmgrd uses the /usr/local/flexlm/licenses/license.dat file. If you omit the *license_file* argument, lmgrd uses the LM_LICENSE_FILE environment variable. If that environment variable is not set, lmgrd uses the /usr/local/flexlm/licenses/license.dat file.

-d Specifies that the host names that are read from the license file should have the local domain name appended to them before sending to a client. This option is useful when clients are accessing licenses from another domain. Available in lmgrd v2.4 and later.

-l *logfile* Specifies the output log file to use; the output log file is sent to stdout by default.

-p Specifies that the lmdown(1) and lmremove(1) commands can be run only by a license administrator; if you use the -p option, the -2 option is required. A *license administrator* is a member of the lmadmin group, or, if the lmadmin group does not exist, a member of group 0. Available in lmgrd v2.4 and later.

-s *interval* Specifies the log file time-stamp interval (in minutes). The default value is 360 minutes.

`-t` *timeout* Specifies the time-out interval (in seconds) during which daemons must complete their connections to each other. The default value is 10 seconds. If the daemons are being run on busy systems or a very heavily loaded network, you may want to use a larger value.

SEE ALSO

`lmdown`(1) for information about shutting down all FLEXlm license daemons gracefully
`lmstat`(1) for information about reporting status of FLEXlm license daemons and feature usage

`license.dat`(5) for information about the license configuration file for FLEXlm licensed applications in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

NAME

`lmhostid` – Displays the host ID of a system

SYNOPSIS

```
lmhostid
```

IMPLEMENTATION

All supported platforms

DESCRIPTION

The `lmhostid` command calls the flexible license manager (FLEXlm) version of `gethostid`(2) and displays the results. In the FLEXlm v2.4 release and later, `lmhostid` is either a link to or a copy of the `lmutil`(1) utility.

The output of `lmhostid` looks like the following display:

```
lmhostid - Copyright (C) 1989-1994 Globetrotter Software, Inc.
The FLEXlm host ID of this machine is "3e9"
```

SEE ALSO

`lmutil`(1) for information about the core FLEXlm utility

NAME

lmremove – Removes specific FLEXlm licenses and returns them to license pool

SYNOPSIS

lmremove [-c *license_file*] *feature user host_name display*

IMPLEMENTATION

All supported platforms

DESCRIPTION

The lmremove command lets the system administrator remove a single user's flexible license manager (FLEXlm) license for a specified feature. This removal might be required if the licensed user was running the software on a node that subsequently crashed. This situation sometimes causes the license to remain unusable. lmremove allows the license to return to the pool of available licenses. In the FLEXlm v2.4 release and later, lmremove is either a link to or a copy of the lmutil(1) utility.

The lmremove command accepts the following arguments:

-c *license_file* Uses the specified *license_file* as input. If you omit the -c option, lmremove uses the LM_LICENSE_FILE environment variable to find the license file to use. If that environment variable is not set, lmremove uses the /usr/local/flexlm/licenses/license.dat file. If you omit the *license_file* argument, lmremove uses the LM_LICENSE_FILE environment variable. If that environment variable is not set, lmremove uses the /usr/local/flexlm/licenses/license.dat file.

feature Specifies the feature name from the FLEXlm license file. You must enter it exactly as the lmstat(1) command displays it.

user Specifies the user name of the license to be removed. You must enter it exactly as the lmstat(1) command displays it.

host_name Specifies the name of the system user from which the user is using the license. You must enter it exactly as the lmstat(1) command displays it.

display Specifies the name of the user's X Windows System display. You must enter it exactly as the lmstat(1) command displays it.

SEE ALSO

lmstat(1) for information about reporting status of FLEXlm license daemons and feature usage
lmutil(1) for information about the core FLEXlm utility

license.dat(5) for information about the license configuration file for FLEXlm licensed applications in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

NAME

`lmreread` – Instructs the FLEXlm license daemon to reread the license file

SYNOPSIS

`lmreread` [`-c` *license_file*]

IMPLEMENTATION

All supported platforms

DESCRIPTION

The `lmreread` command lets the system administrator instruct the flexible license manager (FLEXlm) license daemon to reread the FLEXlm license file, which can be useful if the data in the license file has changed. You can load the new data into the FLEXlm license daemon without shutting down and restarting it.

The `lmreread` command uses the *license_file* from the command line (or the default file if *license_file* is not specified) only to find the license daemon to send it the command to reread the license file. The license daemon always rereads the original file that it loaded. If you must change the path to the license file read by the license daemon, you must shut down the daemon and restart it with the new license file path.

If the server line node names or port numbers have been changed in the license file, you cannot use `lmreread`. In this case, you must shut down the daemon and restart it for the changes to take effect.

The `lmreread` command does not change any option information specified in an options file. If the new license file specifies a different options file, the information is ignored. If you changed your license file and must reread the options file, you must shut down the daemon by using the `lmdown`(1) command and restart it by using the `lmgrd`(1) command. In the FLEXlm v2.4 release and later, `lmreread` is either a link to or a copy of the `lmutil`(1) utility.

The `lmreread` command accepts the following option:

`-c` *license_file* Uses the specified *license_file* as input. If you omit the `-c` option, `lmreread` uses the `LM_LICENSE_FILE` environment variable to find the license file to use. If that environment variable is not set, `lmreread` uses the `/usr/local/flexlm/licenses/license.dat` file. If you omit the *license_file* argument, `lmreread` uses the `LM_LICENSE_FILE` environment variable. If that environment variable is not set, `lmreread` uses the `/usr/local/flexlm/licenses/license.dat` file.

SEE ALSO

`lmdown`(1) for information about shutting down all FLEXlm license daemons gracefully
`lmgrd`(1) for information about starting up FLEXlm license daemons
`lmutil`(1) for information about the core FLEXlm utility

`license.dat`(5) for information about the license configuration file for FLEXlm licensed applications in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

NAME

`lmstat` – Reports status of FLEXlm license daemons and feature usage

SYNOPSIS

`lmstat` [`-a`] [`-A`] [`-c` *license_file*] [`-f` *feature*] [`-l` *regular_expression*] [`-s` *server*] [`-S` *daemon*] [`-t` *timeout*]

IMPLEMENTATION

All supported platforms

DESCRIPTION

The `lmstat` command provides information about the status of the flexible license manager (FLEXlm) license daemon server nodes, vendor daemons, vendor features, and users of each feature. Optionally, you can qualify information by specific server nodes, vendor daemons, or features. In the FLEXlm v2.4 release and later, `lmstat` is either a link to or a copy of the `lmutil`(1) utility.

The `lmstat` command accepts the following options:

`-a` Displays all active users of all features. You should not use the `lmstat -a` command too often because if there are many active users, the `lmstat -a` command can generate a lot of network activity.

`-A` Lists all active licenses.

`-c` *license_file* Uses the specified license file as input. If you omit the `-c` option, `lmstat` uses the `LM_LICENSE_FILE` environment variable to find the license file to use. If that environment variable is not set, `lmstat` uses the `/usr/local/flexlm/licenses/license.dat` file. If you omit the *license_file* argument, `lmstat` uses the `LM_LICENSE_FILE` environment variable. If that environment variable is not set, `lmstat` uses the `/usr/local/flexlm/licenses/license.dat` file.

`-f` *feature* Lists all users of the specified feature.

`-l` *regular_expression* Lists all users of the features who match the specified regular expression.

`-s` *server* Displays the status of the specified server (server node).

`-S` *daemon* Lists all users of the specified daemon's features.

`-t` *timeout* Specifies the time-out interval (in seconds) during which daemons must complete their connections to each other. The default value is 10 seconds. If the daemons are being run on busy systems or a very heavily loaded network, you may want to use a larger value.

SEE ALSO

`lmgrd`(1) for information about starting up FLEXlm license daemons
`lmutil`(1) for information about the core FLEXlm utility

`license.dat`(5) for information about the license configuration file for FLEXlm licensed applications in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

NAME

lmutil – Core FLEXlm utility

SYNOPSIS

lmutil [-c *license_file*] *command*

IMPLEMENTATION

All supported platforms

DESCRIPTION

The lmutil utility is the core flexible license manager (FLEXlm) utility. Usually, end users will not use lmutil directly; they use the individual commands, which are either a copy of or a link to the lmutil utility.

The lmutil utility accepts the following arguments:

-c *license_file* Uses the specified *license_file* as input. If you omit the -c option, lmutil uses the LM_LICENSE_FILE environment variable to find the license file to use. If that environment variable is not set, lmutil uses the /usr/local/flexlm/licenses/license.dat file. If you omit the *license_file* argument, lmutil uses the LM_LICENSE_FILE environment variable. If that environment variable is not set, lmutil uses the /usr/local/flexlm/licenses/license.dat file.

command Links to the specified command. *command* may be lmcksum, lmdown, lmhostid, lmremove, lmreread, lmstat, or lmver.

SEE ALSO

lmcksum(1) for information about computing a checksum for the FLEXlm license file
lmdown(1) for information about shutting down all FLEXlm license daemons gracefully
lmhostid(1) for information about how to display the FLEXlm host ID of a system
lmremove(1) for information about removing specific FLEXlm licenses and returning them to the pool
lmreread(1) for information about instructing the FLEXlm license daemon to reread the FLEXlm license file
lmstat(1) for information about reporting status of FLEXlm license manager daemons and feature usage
lmver(1) for information about how to display the FLEXlm version being used

license.dat(5) for information about the license configuration file for FLEXlm licensed applications in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

NAME

`lmver` – Displays the FLEXlm version being used

SYNOPSIS

`lmver` *filename*

IMPLEMENTATION

All supported platforms

DESCRIPTION

The `lmver` command scans the contents of a binary or library file for the flexible license manager (FLEXlm) version string and displays it. On UNICOS systems, you must use the *filename* operand to display the correct FLEXlm version. If you omit the *filename* operand, `lmver` assumes the file name is `lmgr.a` and tries to find and display the version from the `lmgr.a` file. In the FLEXlm v2.4 release and later, `lmver` is either a link to or a copy of the `lmutil`(1) utility.

The `lmver` command accepts the following operand:

filename Specifies the file to scan to display the FLEXlm version being used. On UNICOS systems, you must use *filename*, and it must be `/usr/lib/libcraylm.a`.

SEE ALSO

`lmutil`(1) for information about the core FLEXlm utility

NAME

ln – Links files

SYNOPSIS

ln [-f] [-m] [-s] *file...* *target*

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4
AT&T extension (-s option)
CRI extension (-m option)

DESCRIPTION

The ln utility creates a link between *file* and *target*. A link is a directory entry that refers to a file; the same file may have several links to it. Under no circumstance can *file* and *target* be the same (take care when using sh(1) metacharacters).

If more than one *file* is specified, *target* must be an existing directory. If *target* is an existing directory, for each specified *file*, a link of the same name is created in the *target* directory. If *target* is not a directory, it is created as a link to *file*. If *target* is an existing file, it will not be overwritten unless the -f option is specified.

The ln utility accepts the following options and operands:

-f Forces files to be linked, even if the target exists. This option works only for hard links.

-m Creates a multilevel symbolic link.

-s Creates symbolic links.

file The path name of a file to be linked.

target The path name of the new link to be created, or the pathname of an existing directory in which the new links are to be created.

There are three kinds of links: hard links, symbolic links, and multilevel symbolic links.

A hard link (the default) can be made only to an existing file. To remove a file with more than one hard link, you must remove all links (including the name by which it was created). Hard links cannot span file systems or refer to directories.

A symbolic link contains the name of the file or directory to which it is linked. Symbolic links may span file systems and may refer to directories.

A multilevel symbolic link is a symbolic link that imposes a multilevel directory structure on any directory to which it points. It works much the same way a symbolic link works, with the exception that it causes the path name search to be deflected into a labeled subdirectory under the directory named in the symbolic link file. Creation of multilevel symbolic links is a privileged operation.

Your active security label must be equal to the file's security label (hard links only) and the parent directory for the new file entry (*target*) (unless the parent directory has a wildcard security level; then a file with any security level may be linked to the wildcard directory).

NOTES

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
`system`, `secadm`	Allowed to link any file or directory.
`sysadm`	Allowed to link any file or directory, subject to security label restrictions. Shell-redirected I/O is subject to security label restrictions.

If the `PRIV_SU` configuration option is enabled, the super user is allowed to link any file or directory.

EXIT STATUS

The `ln` utility exits with one of the following values:

0 All the specified files were linked successfully.

>0 An error occurred.

EXAMPLES

Example 1: The following example links the existing file `example.c` to `ex.c`:

```
$ ln example.c ex.c
```

Example 2: The following example creates a symbolic link:

```
$ ln -s /usr/include incl
$ ls -l incl
lrwxrwxrwx  1  jtk      12 May 10 14:59 incl -> /usr/include
```

SEE ALSO

`cp`(1), `cpio`(1), `mv`(1), `rm`(1), `sh`(1)

`chmod`(2), `link`(2), `readlink`(2), `stat`(2), `symlink`(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

General UNICOS System Administration, Cray Research publication SG–2301

NAME

`locale` – Gets locale-specific information

SYNOPSIS

`locale` [`-a` | `-m`]

`locale` [`-c`] [`-k`] *name* ...

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The `locale` utility writes information about the current locale environment, or all public locales, to the standard output. A *public* locale is one that is accessible to the application.

When you use `locale` without any arguments, it summarizes the current locale environment for each locale category determined by the settings of the `LC_CTYPE`, `LC_COLLATE`, `LC_TIME`, `LC_NUMERIC`, `LC_MONETARY`, and `LC_MESSAGES` environment variables

When you specify a keyword *name*, `locale` selects the named keyword and the category containing that keyword. When a category *name* is specified, `locale` selects the named category and all keywords in that category.

The `locale` utility accepts the following options and operands:

`-a` Writes information about all available public locales. The available locales include `POSIX`, which represents the POSIX locale.

`-m` Writes names of available charmaps.

`-c` Writes the names of selected locale categories.

`-k` Writes the names and values of selected keywords.

name The name of a locale category, the name of a keyword in a locale category, or the reserved name `charmap`. The specified category or keyword is selected for output. You can specify both category and keyword names as *name* operands, in any sequence.

NOTES

If this utility is installed with the default privilege assignment list (PAL), a user with an active `secadm` or `sysadm` category may override mandatory write access protections on a file, any directory in the file path, or, in a privileged administrator shell environment, any file to which input or from which output is being redirected.

EXAMPLES

In the following examples, the assumption is that locale environment variables are set as follows:

```
LANG=locale_x
LC_COLLATE=locale_y
```

Example 1: Invoking `locale` with no arguments results in the following:

```
$ locale
LANG=locale_x
LC_CTYPE="locale_x"
LC_COLLATE=locale_y
LC_TIME="locale_x"
LC_NUMERIC="locale_x"
LC_MONETARY="locale_x"
LC_MESSAGES="locale_x"
LC_ALL=
```

Example 2: Set the `LC_ALL` environment variable to the `POSIX` locale and print out the value of the keyword `decimal_point`:

```
$ LC_ALL=POSIX locale -ck decimal_point
LC_NUMERIC
decimal_point="."
```

EXIT STATUS

The `locale` utility exits with one of the following values:

0 All the requested information was found and output successfully.

>0 An error occurred.

SEE ALSO

`localedef`(1)

General UNICOS System Administration, Cray Research publication SG–2301

NAME

`localedef` – Defines locale environment

SYNOPSIS

`localedef` [`-c`] [`-f` *charmap*] [`-i` *sourcefile*] *name*

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The `localedef` utility converts source definitions for locale categories into a format usable by the functions and utilities whose operational behavior is determined by the setting of the locale environment variables.

The `localedef` utility reads source definitions for one or more locale categories that belong to the same locale from the file specified in the `-i` option (if specified) or from standard output.

Each category source definition is identified by the corresponding environment variable name and terminated by an `END` *category-name* statement. The following categories are supported.

`LC_COLLATE` Defines collation rules.

`LC_CTYPE` Defines character classification and case conversion.

`LC_MESSAGES` Defines the format and values of affirmative and negative responses.

`LC_MONETARY` Defines the format and symbols used in formatting of monetary information.

`LC_NUMERIC` Defines the decimal delimiter, grouping, and grouping symbol for nonmonetary numeric editing.

`LC_TIME` Defines the format and content of date and time information.

The `localedef` utility accepts the following options and operands:

`-c` Creates permanent output even if warning messages have been issued.

`-f` *charmap* Specifies the path name of a file that contains a mapping of character symbols and collating element symbols to actual character encodings. If symbolic names (other than collating symbols defined in a `collating-symbol` keyword) are used, you should specify this option.

`-i` *sourcefile* Specifies the path name of a file that containis the source definitions.

name Identifies the target locale. The utility supports the creation of *public*, or generally accessible locales, as well as *private*, or restricted access locales. If the name contains one or more slash (/) characters, *name* is interpreted as a pathname where the created locale definition(s) will be stored. If the name does not contain any slash characters, the locale will be *public*. The ability to create *public* locales is restricted to users with appropriate privileges. If you omit this option, source definitions are read from standard input.

EXIT STATUS

The `localedef` utility exits with the following value:

3 The capability to create new locales is not supported.

SEE ALSO

`locale`(1)

NAME

logger – Makes entries in the system log

SYNOPSIS

logger [-d] [-f *file*] [-h *host*] [-i] [-l *logname*] [-p *pri*] [-t *tag*] [-P *port*] [-T] [*messages*]

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4
AT&T extensions (-f, -i, -p, and -t options)
CRI extensions (-d, -h, -l, -P, and -T options)

DESCRIPTION

The logger utility provides a program interface to the syslog(3C) system log routine. A message can be given on the command line, which is logged immediately, or a file is read and each line is logged.

The logger utility accepts the following options:

-d Opens the pipe to syslogd(8) without the O_NDELAY flag.

-f *file* Logs the specified file.

-h *host* Sends message to the daemon on the remote host, rather than the daemon on the local machine.

-i Logs the process ID of the log process with each line.

-l *logname* Alternates name for the named pipe interface to the syslog(3C) daemon.

-p *pri* Enters the message with the specified priority (*pri*). The priority may be a number from 0 through 7, or a corresponding word, as follows:

```
0       emerg
1       alert
2       crit
3       err
4       warning
5       notice
6       info
7       debug
```

pri may be preceded by a facility, in the form *facility*.*pri*. Valid facilities include the following:

```
kern            user            mail
daemon          auth            syslog
lpr             local0          local1
local2          local3          local4
local5          local6          local7
```

For a discussion of these facilities, see `syslog`(3C). For example, `-p daemon.info` logs messages as informational (level 6) in the daemon facility. The default is `user.notice`.

`-t` *tag* Marks every line in the log with the specified *tag*.

`-P` *port* Uses the specified TCP/IP port, rather than the one specified in `/etc/services`.

`-T` Uses TCP/IP socket, rather than the named pipe interface to the local `syslog`(3C) daemon.

messages The messages to log. If not specified, the file specified with `-f` or standard input is logged.

EXIT STATUS

The `logger` utility exits with one of the following values:

0 Successful completion.

>0 An error occurred.

EXAMPLES

Example 1: If you restart a daemon in the middle of the day, you could log this event with the following command:

```
$ logger -p user.info restarted development copy of syslog daemon
```

The message is as follows:

```
restarted development copy of syslog daemon
```

Example 2: If you are a system operator and have cleaned out files in `/tmp`, you could log this event in the system log maintained by `syslogd`(8), using the following command:

```
$ logger /tmp was cleaned out by hand when it filled up.
Joe Operator
```

The message is as follows:

```
/tmp was cleaned out by hand when it filled up.
Joe Operator
```

SEE ALSO

`syslog`(3C) in the *UNICOS System Libraries Reference Manual*, Cray Research publication SR–2080

`log`(4) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

`syslogd`(8) in the *UNICOS Administrator Commands Reference Manual*, Cray Research publication SR–2022

NAME

login – Signs on

SYNOPSIS

login [*name* -L *requested_label* [*env-vars*]]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The login utility is used at the beginning of each terminal session and lets you identify yourself to the system. It is invoked by the system when a connection is first established.

Also, it is invoked by the system when a previous user has terminated the initial shell by pressing <CONTROL-d> to indicate an end-of-file.

The login utility invokes the centralized identification and authorization library routines to validate the user ID and password.

login accepts the following options:

name Your login name.

-L *requested_label*
Security label with which you want to log in. The format of the requested label is *level*[,*compartment*[,*compartment*]]. The *requested_label* must immediately follow the *name*.

env-vars Environment variable that you can set.

The login command requests your user name (if not supplied as an argument), and, if appropriate, your password. Echoing is turned off (where possible) during the typing of your password, so it will not appear on the written record of the session.

After a successful login, accounting files are updated, the procedure /etc/profile is performed, the message-of-the-day, if any, is printed, the user ID, the group ID, the working directory, and the command interpreter (usually sh(1)) are initialized, and the file .profile in the working directory is executed, if it exists. In addition, if executing in a secure UNICOS environment, the user's default and maximum class, default and authorized categories, default and authorized compartments, minimum and maximum security levels, default security level, and security permissions are set. These specifications are found in the user's /etc/udb file entry. The name of the command interpreter is - followed by the last component of the interpreter's path name (that is, -sh). If this field in the udb file is empty, the default command interpreter, /bin/sh, is used. If this field is *, a chroot(2) is performed to the directory named in the directory field of the entry. At that point, login is reexecuted at the new level, which must have its own root structure, including /bin/login and /etc/udb.

The basic environment (see sh(1)) is initialized to the following:

HOME=*your-login-directory*
PATH=:/bin:/usr/bin:/usr/ucb
SHELL=*last-field-of-udb-ue_shell field*
MAIL=/usr/mail/*your-login-name*
LOGNAME=*your-login-name*

You may modify the environment by supplying additional arguments to login, either at execution time or when login requests your login name. The *env-vars* arguments may take either the form *xxx* or *xxx=yyy*. Arguments without an equal sign are placed in the environment as follows:

L*n*=xxx

The *n* argument is a number starting at 0 and is incremented each time a new variable name is required. Variables containing = are placed into the environment without modification. If they already appear in the environment, they replace the older value. There are two exceptions: the variables PATH and SHELL cannot be changed. Users logging into restricted shell environments are thus prevented from spawning secondary shells that are not restricted.

The login utility understands simple single-character quoting conventions. Typing a backslash (\) in front of a character quotes it and allows the inclusion of such characters as spaces and tabs.

NOTES

It is not possible to exec(2) login from a normal user ID.

The number of unsuccessful login attempts that will be allowed before login terminates is configurable. This parameter is set in the /etc/config/confval file with the following line:

```
login.login_attempts:   "3"
```

"3" is the number of unsuccessful login attempts allowed. This line is retrieved with the getconfval(3C) library call. If this line does not exist (the default) or the number is set to 0, no limit will be placed on the number of unsuccessful login attempts.

If the IP security option is not enabled, login requests are validated to ensure that the remote host or workstation's security levels and compartments as defined in the /etc/config/spnet.conf file are included in the security level range and authorized compartment range for the UNICOS system. Your security values are set to the most restrictive boundary conditions as defined by the network access list (NAL) and the user database (UDB).

If the IP security option is enabled, the checks for the /etc/config/spnet.conf file are done by the kernel. The socket's security label is set by the kernel when this check is done. login sets the user's security label to the most restrictive boundary conditions as defined by the socket's security label (as defined by the NAL) and the UDB.

Your active security level and active compartments are obtained from the dev/ttyp file, which contains the security label present with an IP security option. If dev/ttyp has a null security label, the user session is restricted to operating with a null security label, and you are not allowed to change the active security level and active compartments set by login.

The login process also validates your right to access the UNICOS system from the host or workstation issuing the login request. Access to UNICOS is granted or denied based upon the workstation access list (WAL) check performed by the login process.

You cannot log in if you exceed the MAXLOGS setting.

The results of user validation are recorded in the security log.

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
system	Allowed to use this utility.

If the PRIV_SU configuration option is enabled, the super user is allowed to use this utility.

MESSAGES

Login incorrect
: Your name and the password may not match. This is a generic message for any of several login failure causes. No more information is given. The user name may be invalid or the password may be wrong. You may have been denied access by the WAL check. Your login can be locked, disabled, or invalidated for security violations.

No shell
Bad account ID
Bad group ID list
Bad user ID
Unable to change to login root
No Root Directory
: Account may not be set up correctly; consult a system administrator.

No utmp entry
: You must execute login from the lowest-level shell. You attempted to execute login as a command without using the shell's exec internal command or from other than the initial shell.

Unable to give N shares to user
: The setshares call failed. Contact your system support staff.

Unable to create new job
: The setjob call failed. Contact your system support staff.

Unable to make job temporary directory
: The makejtmp call failed. Contact your system support staff.

`Unable to lock the UDB`
: The `lockudb` call failed. Contact your system support staff.

`Unable to update the UDB`
: The rewrite `udb` call failed while attempting to write the last login record. Contact your system support staff.

`Lastlog update error`
: Unable to reread the `udb` entry to write the last login record. Contact your system support staff.

`Login not allowed at this node`
: You are not allowed to log in at this network node for security reasons. Use an authorized terminal for logging in.

`Unable to get host by name`
: The host name that accompanies the login request is not defined in `/etc/hosts` or is *null*.

`Could not access NAL`
: An error was detected when `/etc/config/spnet.conf` was opened. Contact your system support staff.

`No login without NAL entry`
: The user does not have an NAL entry for this remote node.

`Can't set default security level`

`Can't set default security compartments`

`urm: job exceeds memory maximum`
: The job would exceed the memory oversubscription amount configured in the Unified Resource Manager (URM).

`urm: job exceeds job maximum`
: The job would exceed the number of allowed jobs configured in the Unified Resource Manager (URM).

`Invalid requested label`
: The requested label is not valid.

FILES

`/bin/passwd`	Program that changes passwords
`/bin/sh`	Standard shell
`/dev/tty*`	Login devices
`/etc/dialups`	List of devices that need a dial-up password
`/etc/d_passwd`	Dial-up passwords for `/etc/dialups`
`/etc/utmp`	Accounting file

`/etc/wtmp`	Accounting file
`/usr/mail/$LOGNAME`	Mailbox for account `$LOGNAME`
`/etc/udb`	User validation file containing user control limits
`/etc/config/confval`	Number of bad login attempts after which `login` terminates
`/etc/config/spnet.conf`	Network access list (NAL) and workstation access list (WAL)

SEE ALSO

`mail`(1), `passwd`(1), `sh`(1), `su`(1)

`chroot`(2), `exec`(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

`getconfval`(3C), `ia_failure`(3C), `ia_mlsuser`(3C), `ia_success`(3C), `ia_user`(3C) in the *UNICOS System Libraries Reference Manual*, Cray Research publication SR–2080

`cshrc`(5), `profile`(5), `udb`(5) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

`chroot`(8), `getty`(8) in the *UNICOS Administrator Commands Reference Manual*, Cray Research publication SR–2022

`checkwal`(), `fetchnal`() (library routines in `/libc/gen`)

General UNICOS System Administration, Cray Research publication SG–2301

NAME

logname – Gets user's login name

SYNOPSIS

logname

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The logname utility uses getlogin(3C) to find the login name of the user and writes that name to standard output.

EXIT STATUS

The logname utility exits with one of the following values:

0 Successful completion.

>0 An error occurred.

SEE ALSO

env(1), login(1), sh(1)

getlogin(3C) in the *UNICOS System Libraries Reference Manual*, Cray Research publication SR–2080

NAME

`lorder` – Finds ordering relation for an object library

SYNOPSIS

`lorder` *files*

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The input is one or more object or library archive *files* (see `ar`(1)). The standard output is a list of pairs of object file or archive member names, meaning that the first file of the pair refers to external identifiers defined in the second. The output may be processed by `tsort`(1) to find an ordering of a library suitable for one-pass access by `ld`(1).

Link editor `ld`(1) is capable of multiple passes over an archive in the portable archive format (see `ar`(5)) and does not require that `lorder` be used when building an archive. The usage of the `lorder` utility may, however, allow for a slightly more efficient access of the archive during the link edit process. The use of this utility is not recommended.

The `lorder` utility accepts the following option:

files Names of object or library archive files you specify.

NOTES

When given a nonexistent file, `lorder` returns an exit status of 0.

WARNINGS

The `lorder` utility accepts as input any object or archive file, regardless of its suffix, provided that there is more than one input file. If there is only one input file, its suffix must be `.o`.

EXAMPLES

The following example builds a new library from existing `.o` files:

```
bar cr library ‘lorder *.o | tsort‘
```

FILES

`TMPDIR/*symdef` Temporary file

`TMPDIR/*symref` Temporary file

SEE ALSO

`ar`(1), `ld`(1), `tsort`(1)

`tmpnam`(3C) in the *UNICOS System Libraries Reference Manual*, Cray Research publication SR–2080

`ar`(5) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

`tsar`(8) in the *UNICOS Administrator Commands Reference Manual*, Cray Research publication SR–2022

NAME

lp – Sends files to a printer

SYNOPSIS

lp [-c] [-d *dest*] [-n *copies*] [-s] [*file...*]

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The lp utility copies the input files to an output device. The actual writing to the output device occurs after the lp utility successfully exits.

The lp utility accepts the following options:

-c Exits only after further access to any of the input files is no longer required. The application can then safely delete or modify the files without affecting the output operation.

-d *dest* Specifies a string that specifies the output device or destination. The -d *dest* option takes precedence over the LPDEST environment variable, which in turn takes precedence over the PRINTER environment variable.

-n *copies* Writes *copies* number of copies of the files; *copies* is a positive decimal integer.

-s Suppress printing Request id is . . . message to standard output.

file Denotes a path name of a file to be output. If you omit *file* operands, or if a *file* operand is -, the standard input is used. If a *file* operand is used, but you omit -c option, the process performing the writing to the output device may have user and group permissions that differ from that of the process invoking lp.

ENVIRONMENT VARIABLES

LPDEST This variable is interpreted as a string that names the output device or destination. If the LPDEST environment variable is not set, the PRINTER environment variable is used. The -d *dest* option takes precedence over LPDEST.

PRINTER This variable is interpreted as a string that names the output device or destination. If the LPDEST and PRINTER environment variables are not set, a system default printer is used. The -d *dest* and the LPDEST environment variable take precedence over PRINTER.

EXIT STATUS

The `lp` utility exits with one of the following values:

0 All input files were processed successfully.

>0 No output device was available, or an error occurred.

SEE ALSO

`lpr`(1B), `sh`(1)

NAME

lpq – Spool queue examination program

SYNOPSIS

/usr/ucb/lpq [+[*n*]] [-l] [-P*printer*] [*jobnums*] [*users*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The lpq utility examines the spooling area used by lpd(8) for printing files on the line printer, and reports the status of the specified jobs or all jobs associated with a user. If you invoke lpq without any arguments, it reports on any jobs currently in the queue. The lpq utility prints only those jobs at the current security label.

The lpq utility accepts the following options:

+[*n*] Displays the spool queue. When *n* (a number) is specified, lpq sleeps for *n* seconds between scans of queue.

-l Causes information for each file comprising a job to be printed.

-P*printer* Specifies a particular printer. When P is not specified, either the default line printer or the value of the PRINTER variable in the environment is used.

jobnums Specifies job numbers that should be examined.

users Specifies user names that should be examined.

For each job submitted (that is, each invocation of lpr(1B)), lpq reports the user's name, the current rank in the queue, the names of files that compose the job, the job identifier (a number that may be supplied to lprm(1B) for removing a specific job), and the total size (in bytes). Unless the -l option is used, only as much information as will fit on one line is displayed. Job ordering depends on the algorithm used to scan the spooling directory and is supposed to be FIFO (first-in, first-out). File names composing a job may be unavailable (when lpq is used as a sink in a pipeline), in which case, the file is indicated as standard input.

If lpq warns that no daemon present (that is, because of some malfunction), the lpc(8) command can be used to restart the printer daemon.

NOTES

If users try to remove files other than their own, permission will be denied.

If this utility is installed with a privilege assignment list (PAL), a user who is assigned the following privilege text upon execution of this command is allowed to perform the action shown:

Privilege Text	Action
admin	Allowed to see the status of all jobs in a print queue. Other users may only see jobs in the print queue at the same label as the user.

If this command is installed with a PAL, a user with one of the following active categories is allowed to perform the action shown:

Active Category	Action
sysadm, sysadm	Allowed to see the status of all jobs in a print queue.

If the PRIV_SU option is enabled, the super user is allowed to see all jobs in the print queue.

BUGS

The output of lpq may be somewhat unreliable because of the dynamic nature of the information in the spooling directory.

Output formatting is sensitive to the line length of the terminal; this can result in widely spaced columns.

The lpq utility is sometimes unable to open various files because the lock file is malformed.

FILES

/etc/printcap	To determine printer characteristics
/etc/termcap	To manipulate the screen for repeated display
/usr/spool/*	Spooling directory, as determined from printcap
/usr/spool/*/cf*	Control files specifying jobs
/usr/spool/*/lock	Lock file to obtain the currently active job

SEE ALSO

lpr(1B), lprm(1B)

printcap(5) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

lpc(8), lpd(8) in the *UNICOS Administrator Commands Reference Manual*, Cray Research publication SR–2022

NAME

lpr – Prints off-line

SYNOPSIS

/usr/ucb/lpr [-P*printer*] [-#*num*] [-C *class*] [-J *job*] [-T *title*] [-R] [-i[*numcols*]] [-1|2|3|4*font*] [-w*num*] [-p] [-l] [-t] [-n] [-d] [-g] [-v] [-c] [-f] [-r] [-m] [-h] [-s] [*files*]

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The lpr utility uses a spooling daemon to print the files specified by *files* when facilities become available. If no files are specified, standard input is assumed.

The lpr utility accepts the following option to specify a printer:

-P*printer* Forces output to a specific printer. Usually, the default printer is used (site-dependent), or the value of the PRINTER environment variable is used.

The following single-letter options are used to notify the line printer spooler that the files are not standard text files. The spooling daemon uses the appropriate filters to print the data accordingly.

-p Uses pr(1) to format the files (equivalent to print).

-l Uses a filter that allows control characters to be printed and suppresses page breaks.

-t Assumes that the files contain data from troff (cat phototypesetter commands).

-n Assumes that the files contain data from ditroff (device-independent troff).

-d Assumes that the files contain data from tex (DVI format from Stanford).

-g Assumes that the files contain standard plot data as produced by the plot routines (for the filters used by the printer spooler).

-v Assumes that the files contain a raster image for devices such as the Benson Varian.

-c Assumes that the files contain data produced by cifplot.

-f Uses a filter that interprets the first character of each line as a standard Fortran carriage control character.

You can use the following options:

`-#`*num* Prints multiple copies of output; *num* is the number of copies desired of each file specified. For example, the following command line would result in three copies of file `foo.c`, followed by three copies of file `bar.c`, and so on.

```
lpr -#3 foo.c bar.c more.c
```

On the other hand, the following command line would produce three copies of the concatenation of the files:

```
cat foo.c bar.c more.c | lpr -#3
```

`-C` *class* Uses *class* arguments as a job classification for use on the burst page. For example, the following command line would cause the system name (the name returned by `hostname`(1)) to be replaced on the burst page by `EECS`, and file `foo.c` to be printed:

```
lpr -C EECS foo.c
```

`-J` *job* Uses *job* as the job name to print on the burst page. Usually, the name of the first file is used.

`-T` *title* Uses *title* as the title used by `pr`(1), instead of the file name.

`-R` Writes a message to standard output containing the unique number which is used to identify this job. This number can be used to cancel (see `lprm`(1B)) or find the status (see `lpq`(1B)) of the job.

`-i`[*numcols*]
Indents the output. If *numcols* is numeric, it will be used as the number of blanks to be printed before each line; otherwise, 8 characters are printed.

`-1|2|3|4`*font*
Specifies a font to be mounted on font position 1, 2, 3, or 4. The daemon constructs a `.railmag` file referencing file `/usr/lib/vfont/name.size`.

`-w`*num* Sets the page width to *num*.

`-r` Removes the file on completion of spooling or on completion of printing (with the `-s` option).

`-m` Sends mail on completion.

`-h` Suppresses the printing of the burst page.

`-s` Links data files rather than trying to copy them so that large files can be printed.

files Files to be printed. This means that the files must not be modified or removed until they have been printed.

NOTES

If you try to spool a file that is too large, it will be truncated.

lpr objects to printing binary files.

If a user other than root prints a file and spooling is disabled, lpr will print a message saying this and will not put jobs in the queue.

If a connection to lpd(8) on the local machine cannot be made, lpr will indicate that the daemon cannot be started. Messages can be printed in the daemon's log file regarding missing lpd spool files.

BUGS

Fonts for troff and tex reside on the host with the printer. Currently, local font libraries cannot be used.

FILES

/etc/printcap	Printer capabilities database
/etc/udb	User validation file containing user control limits
/usr/lib/lpd*	Line printer daemons
/usr/lib/lpf	Sample lpr output filtering program
/usr/lib/necf	Sample lpr output filtering program
/usr/spool/*	Directories used for spooling
/usr/spool/*/cf*	Daemon control files
/usr/spool/*/df*	Data files specified in cf files
/usr/spool/*/tf*	Temporary copies of cf files

SEE ALSO

lp(1), lpq(1B), lprm(1B), pr(1)

lpc(8), lpd(8) in the *UNICOS Administrator Commands Reference Manual*, Cray Research publication SR–2022

NAME

`lprm` – Removes jobs from the line printer spooling queue

SYNOPSIS

`/usr/ucb/lprm` [`-P`*printer*] [`-`] [*jobnums*] [*users*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `lprm` utility removes a job, or jobs, from a printer's spool queue. Because the spooling directory is protected from users, using `lprm` is typically the only method by which you may remove a job.

If you specify `lprm` without any arguments, it deletes the currently active job if it is owned by the user who invoked `lprm`.

The `lprm` utility accepts the following options:

`-P`*printer* Specifies the queue associated with a specific printer; otherwise, the default printer or the value of the `PRINTER` environment variable is used.

– Removes all jobs that a user owns. If the super user uses this option, the spool queue will be emptied entirely.

jobnums Used to dequeue an individual job.

users Removes any jobs queued belonging to the user (or users).

You may dequeue an individual job by specifying its job number. This number may be obtained from the `lpq`(1B) program. For example, if you use the following program, `lprm` will announce the names of any files it removes; it will remain silent if there are no jobs in the queue that match the request list:

```
$ lpq -l
1st: ken        [job #013ucbarpa]
        (standard input)        100 bytes
$ lprm 13
```

The `lprm` utility kills an active daemon, if necessary, before removing any spooling files. If a daemon is killed, a new one is automatically restarted upon completion of file removal.

NOTES

If users try to remove files other than their own, permission will be denied.

If this utility is installed with a privilege assignment list (PAL), a user who is assigned the following privilege text upon execution of this command is allowed to perform the action shown:

Privilege Text	Action
admin	Allowed to remove any job in a print queue. Other users may only remove their own jobs.

If this command is installed with a PAL, a user with one of the following active categories is allowed to perform the action shown:

Active Category	Action
sysadm, sysadm	Allowed to remove any job in a print queue.

If the PRIV_SU option is enabled, the super user is allowed to remove any job in a print queue.

BUGS

Because race conditions are possible in the update of the lock file, the currently active job may be incorrectly identified.

FILES

/usr/spool/*	Spooling directories
/etc/printcap	Printer characteristics file
/usr/spool/*/lock	Lock file used to obtain the process identification number (PID) of the current daemon and the job number of the currently active job

SEE ALSO

lpq(1B), lpr(1B)

NAME

`ls` – Lists contents of directory

SYNOPSIS

`ls` [`-1`] [`-a`] [`-A`] [`-b`] [`-B`] [`-c`] [`-C`] [`-d`] [`-e`] [`-f`] [`-F`] [`-g`] [`-i`] [`-k`] [`-I`] [`-l`] [`-L`] [`-m`] [`-n`] [`-o`] [`-p`] [`-P`] [`-q`] [`-r`] [`-R`] [`-s`] [`-t`] [`-u`] [`-x`] [*file* ...]

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4
AT&T extensions (`-b` option)
CRI extensions (`-A`, `-B`, `-e`, `-k`, `-l`, and `-P` options)

DESCRIPTION

For each directory argument, `ls` lists the contents of the directory; for each file argument, `ls` repeats the file name and any other information requested. By default, the output is sorted alphabetically. When no argument is specified, the current directory is listed. When several arguments are specified, the arguments are first sorted appropriately, but file arguments appear before directories and their contents.

There are three major listing formats. The default format for output directed to a terminal is multi-column with entries sorted down the columns. The `-1` option allows single-column output, and the `-m` option enables stream output format in which files are listed across the page, separated by commas.

To determine output formats for the `-C`, `-x`, and `-m` options, `ls` uses the `COLUMNS` environment variable, which determines the number of character positions available on one output line. If this variable is not set, the system is prompted by using the `ioctl`(2) system call to acquire the current window size. When this information cannot be obtained, 80 columns are assumed.

Use the `-e` option to display security information.

The `ls` utility accepts the following options:

`-1` (Number one) Forces output format of one entry per line. This is the default format when the output is not directed to a terminal.

`-a` Lists all entries, including entries whose names begin with a dot (`.`), which usually are not listed.

`-A` Lists all entries, including entries whose names begin with a dot (`.`), except for the dot (`.`) and dot-dot (`..`) files.

`-b` Forces the printing of nongraphic characters to be in the octal *ddd* notation.

`-B` Writes the number of file system blocks a file occupies, as specified by the `-l` and `-s` options, in 4096-byte units, rather than the default 512-byte units.

`-c` Uses time of last modification of the inode (file created, mode changed, and so on) for sorting (`-t`) or printing (`-l`).

`-C` Specifies multicolumn output with entries sorted down the columns. This is the default format when the output is directed to a terminal.

`-d` If the *files* argument is a directory, lists only its name (not its contents); often used with `-l` to get the status of a directory.

`-e` Provides the access control list (ACL) flag, integrity label flag, security level, and compartment flag as the last fields before the file name, or, when used with `-l`, as the fields immediately following the mode field. When at least one file compartment is set, the file's compartment flag is displayed as a plus sign (`+`) adjacent to the file's security level. An `i` indicates that the file has an integrity class or category assigned to it; ask your security administrator to relabel these values to 0. When a file has an associated ACL, its ACL flag appears as an `a` adjacent to the mode field (if present), or following any other selected fields. A file or directory that has a wildcard security level has an asterisk (`*`) displayed for its file level. A trusted facility management file or directory has a `T` displayed for its file level; if this symbol is displayed, ask your security administrator to relabel the file with the proper security label. The file's security level is displayed as a question mark (`?`) if it is outside the user's allowable range, or if any of the file's compartments are not in the user's set of valid compartments. No security information is displayed when a question mark is shown.

`-f` Forces each argument to be interpreted as a directory and lists the name found in each slot. This option turns off `-A`, `-l`, `-t`, `-s`, and `-r`, and turns on `-a`; the order is the order in which entries appear in the directory.

`-F` Adds a slash (`/`) after each file name that is a directory, adds an asterisk (`*`) after each file name that is executable, adds an at sign (`@`) after each file name that is a symbolic link, adds an equals sign (`=`) after each file name that is a socket, and a vertical line (`|`) after each file name that is a FIFO.

`-g` The same as `-l`, except that the owner is not printed.

`-i` For each file, prints the inode number in the first column of the report.

`-k` Writes the number of file system blocks a file occupies, as specified by the `-l` and `-s` options, in 1024-byte units, rather than the default 512-byte units.

`-I` Same as `-i`, but the inode is printed as two 32-bit values.

`-l` Lists in long format, specifying mode, number of links, owner, group, size in bytes, and time of last modification for each file. See the EXAMPLES section. If the file is a special file, the size field will contain the major and minor device numbers rather than a size. If the file is a symbolic link, the path name of the linked-to file will be printed, preceded by `->`; or, if the file is a multilevel symbolic link and the `-e` option is used, the pathname will be preceded by `*>`. If *file* is a directory, each list of files within the directory is preceded by a status line that indicates the number of file system blocks occupied by files in the directory in 512-byte units (rounded up, if necessary).

`-L` If an argument is a symbolic link, lists the file or directory the link references rather than the link itself.

`-m` Specifies stream output format.

`-n` The same as `-l`, except that the owner's UID and GID numbers (and account ID numbers for `-P`), rather than the associated character strings, are printed.

`-o` The same as `-l`, except that the group is not printed.

`-p` Puts a slash (`/`) after each file name if that file is a directory.

`-P` Lists the account ID associated with each file.

`-q` Forces printing of nongraphic characters in file names as the character `?`.

`-r` Reverses the order of sort to get reverse alphabetic or oldest first, as appropriate.

`-R` Recursively lists subdirectories encountered.

`-s` Gives size of each file in terms of file system blocks. The size is given in 512-byte units by default, and may be changed by the `-k` or `-B` options.

`-t` Sorts by time modified (latest first) rather than by name.

`-u` Uses time of last access rather than last modification for sorting (with the `-t` option) or printing (with the `-l` option).

`-x` Specifies multi-column output with entries sorted across rather than down the page.

files Files to be listed. If no *files* are specified, all files in the current directories are listed.

The mode printed under the `-l` option consists of 10 characters. The first character is one of the following:

`b` Block special file
`c` Character special file
`d` Directory.
`l` Symbolic link
`m` Migrated file
`p` FIFO (named pipe) special file
`R` Restart file; for more information on the restartability of core files, see `restart`(1) and `core`(5).
`s` Type socket
`-` Ordinary file

The next 9 characters are interpreted as three sets of 3 bits each. The first set refers to the owner's permissions, the next to permissions of others in the user-group of the file, and the last to all others. Within each set, the three characters indicate permission to read, to write, and to execute the file as a program, respectively. For a directory, execute permission is interpreted as permission to search the directory for a specified file. The permissions are indicated as follows:

`r` The file is readable.
`w` The file is writable.
`x` The file is executable.

- `-` The indicated permission is not granted.
- `l` Mandatory locking occurs during access (the set-group-ID it is on, and the group execution bit is off).
- `t` Sets the sticky bit (see `chmod`(2) for more information). Only the super user or owner of the directory can alter the `t` permission (mode 1000).
- `s` The set-user-ID or set-group-ID bit is on, and the corresponding user or group execution bit is also on.
- `S` Undefined bit-state (the set-user-ID bit is on and the user execution bit is off).

For user and group permissions, the third position is sometimes occupied by a character other than `x` or `-`. `s` may also occupy this position; it refers to the state of the set-ID bit, whether it be the user's or the group's. The ability to assume the same ID as the user during execution is, for example, used during login when you begin as `root` but must assume the identity of the user stated at login.

In the case of the sequence of group permissions, `l` may occupy the third position. `l` refers to mandatory file and record locking. This permission describes a file's ability to allow other processes to lock its reading or writing permissions during access.

When the sizes of the files in a directory are listed, a total count of blocks, including indirect blocks, is printed.

If you specify the `-l` option, each list of files within the directory is preceded by a status line indicating the number of file system blocks occupied by files in the directory in 512-byte units.

NOTES

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
`system`, `secadm`	In a privileged administrator shell environment, shell-redirected I/O is not subject to file protections.
`sysadm`	Shell-redirected output is subject to security label restrictions.

If the `PRIV_SU` configuration option is enabled, shell-redirected I/O on behalf of the super user is not subject to file protections.

EXIT STATUS

The `ls` utility exits with one of the following values:

0 All files were written successfully.

>0 An error occurred.

BUGS

Unprintable characters in file names may confuse the columnar output options.

In file names, <newline> and <tab> are considered printing characters.

The output device is assumed to be 80 columns wide.

If hard links are among the files, the total block count will be incorrect.

EXAMPLES

Example 1: The following is an example of output from the ls utility, using the -l option:

```
total 155
drwxr-xr-x   2 cray     os          96  Sep 23  12:42  onea
-rw-r--r--   1 cray     os        3380  Oct  3  08:18  save.ftpl
-rwx------   1 cray     os       74912  Sep 19  09:04  testx
```

Example 2: The following is an example of output from the ls utility using the -le options. The second column that follows the mode field shows the file's security level. The plus sign (+) appended to the security level of the second file indicates that the file has compartments. The third file has an associated ACL, as indicated by the a in column 1 adjacent to the mode field.

```
total 155
drwxr-xr-x     1    2 cray   os      96   Sep 23   12:42   onea
-rw-r--r--     1+   1 cray   os    3380   Oct 03   08:18   save.ftpl
-rwx------a    1    1 cray   os   74912   Sep 19   09:04   testx
```

FILES

/etc/udb	User validation file that contains user control limits
/etc/group	Group file that contains group names and group IDs

SEE ALSO

chmod(1), find(1), restart(1), chmod(2), socket(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

core(5) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

NAME

m4 – Invokes a macro processor

SYNOPSIS

m4 [-B *int*] [-D *name*[=*val*]] [-e] [-H *int*] [-S *int*] [-s] [-T *int*] [-U *name*] [*files*]

IMPLEMENTATION

All Cray Research systems

STANDARDS

XPG4

DESCRIPTION

The m4 utility invokes a macro processor intended for use as a general-purpose front end for any programming language. Each of the argument files is processed in order; if there are no files, or if a file name is -, standard input is read. The processed text is written on standard output.

The m4 command supports the following options:

-B *int*	Changes the size of the push-back and argument collection buffers from the default of 4096.
-D *name*[=*val*]	Defines *name* to *val* or to null in the absence of *val*.
-e	Operates interactively. Interrupts are ignored and the output is unbuffered.
-H *int*	Changes the size of the symbol table hash array from the default of 199. The size should be a prime number.
-S *int*	Changes the size of the call stack from the default of 100 slots.
-s	Enables line sync output for the C preprocessor (# "line . . .").
-T *int*	Changes the size of the token buffer from the default of 512 bytes.
-U *name*	Undefines *name*. Macros take 3 slots, and nonmacro arguments take 1.
files	Specifies the files to be processed.

Macro calls have the following form:

name(*arg*1,*arg*2, . . ., *argn*)

A (character must immediately follow the name of the macro. When the name of a defined macro is not followed by (, it is considered a call of that macro with no arguments. Potential macro names consist of alphabetic letters, digits, and underscores; the first character cannot be a digit.

Leading unquoted blanks, tabs, and new-line characters are ignored during the collection of arguments. Left single quotation marks (grave accent, ASCII 96) and right single quotation marks are used to quote strings. The value of a quoted string is the string stripped of the quotation marks.

When a macro name is recognized, its arguments are collected by a search for a matching right parenthesis. If fewer arguments are supplied than are in the macro definition, the trailing arguments will be considered null. Macro evaluation proceeds normally during the collection of the arguments, and any commas or right parentheses that appear in the value of a nested call are as effective as those in the original input text. After argument collection, the value of the macro is pushed back into the input stream and rescanned.

The `m4` command has the following built-in macros. They may be redefined, but after redefinition the original meaning is lost. Their values are null unless otherwise stated.

`define` The second argument is installed as the value of the macro whose name is the first argument. Each occurrence of `$`*n* in the replacement text, where *n* is a digit, is replaced by the *n*th argument. Argument 0 is the name of the macro; missing arguments are replaced by the null string; `$#` is replaced by the number of arguments; `$*` is replaced by a list of all arguments separated by commas; `$@` is like `$*`, but each argument is quoted (with the current quotation characters).

`undefine` Removes the definition of the macro named in its argument.

`defn` Returns the quoted definition of its arguments. This is useful for renaming macros, especially built-in macros.

`pushdef` Functions as does `define`, but saves any previous definition.

`popdef` Removes current definition of its arguments, exposing the previous argument if one exists.

`ifdef` If the first argument is defined, the value will be the second argument; otherwise it will be the third. If there is no third argument, the value will be null. The words `unix` and `CRAY` are predefined on all UNICOS systems. The word `CRAYYMP` is predefined.

`shift` Returns all but its first argument. The other arguments are quoted and pushed back, with commas to separate them. The quoting nullifies the effect of the extra scan that will subsequently be performed.

`changequote` Changes quote symbols to the first and second arguments. The symbols may be up to 5 characters long. `changequote` without arguments restores the original values (that is, ‘ ’).

`changecom` Changes left and right comment markers from the default # and new-line character. Without arguments, the comment mechanism is effectively disabled. With one argument, the left marker becomes the argument and the right marker becomes a new-line character. With two arguments, both markers are affected. Comment markers may be up to 5 characters long.

`divert`	Changes the current output stream to its (digit-string) argument. Output diverted to a stream other than 0 through 9 is discarded. The final output is the concatenation of the streams in numeric order; initially stream 0 is the current stream. The `divert` macro maintains 10 output streams, numbered 0 through 9.
`undivert`	Causes immediate output of text from diversions named as arguments, or from all diversions if no argument exists. Text may be undiverted into another diversion. Undiverting discards the diverted text.
`divnum`	Returns the value of the current output stream.
`dnl`	Reads and discards characters up to and including the next new-line character.
`ifelse`	Has three or more arguments. If the first argument is the same string as the second, the value will be the third argument. If not, and if there are more than four arguments, the process will be repeated with arguments four, five, six, and seven. Otherwise, the value will be either the fourth string or, if it is not present, null.
`incr`	Returns the value of its argument incremented by 1. The value of the argument is calculated by the interpreting of an initial digit-string as a decimal number.
`decr`	Returns the value of its argument decremented by 1.
`eval`	Evaluates its argument as an arithmetic expression, using 32-bit arithmetic. Operators include `+`, `-`, `*`, `/`, `%`, bitwise `&`, `|`, `^`, and `~`; relationals; and parentheses. Octal and hexadecimal numbers may be specified as in C. The second argument specifies the radix for the result; the default is 10. The third argument may be used to specify the minimum number of digits in the result.
`len`	Returns the number of characters in its argument.
`index`	Returns the position in its first argument at which the second argument begins (zero origin), or –1 if the second argument does not occur.
`substr`	Returns a substring of its first argument. The second argument is a zero origin number selecting the first character; the third argument indicates the length of the substring. A missing third argument is taken to be large enough to extend to the end of the first string.
`translit`	Transliterates the characters in its first argument from the set given by the second argument to the set given by the third. No abbreviations are permitted.
`include`	Returns the contents of the file named in the argument.
`sinclude`	Functions as does `include`, except that it returns nothing if the file is inaccessible.
`syscmd`	Executes the UNICOS command given in the first argument. No value is returned.
`sysval`	Returns code from the last call to `syscmd`.
`maketemp`	Fills in a string of *XXXXX* in its argument with the current process ID.
`m4exit`	Causes immediate exit from `m4`. Argument 1, if provided, is the exit code; the default is 0.

`m4wrap`	Pushes back argument 1 at final EOF; for example, `m4wrap(cleanup())`.
`errprint`	Prints its argument on the diagnostic output file.
`dumpdef`	Prints current names and definitions, either for the named items or for all, if no arguments are specified.
`traceon`	Without arguments, turns on tracing for all macros (including built-in macros); otherwise turns on tracing for named macros.
`traceoff`	Turns off trace globally and for any macros specified. Macros specifically traced by `traceon` can be untraced only by specific calls to `traceoff`.

SEE ALSO

`cc`(1)

NAME

machid – Gives truth value about processor type

SYNOPSIS

```
crayxmp
crayymp
crayympe
crayympel
crayc90
crayj90
crayts
crayt3d
crayt3e
pdp11
sparc
sun
u370
u3b
u3b5
vax
```

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The machid utility gives the truth value about the processor type.

EXIT STATUS

The following will return a true value (exit code of 0) if you are on a processor that the command name indicates:

crayc90	True if you are on a CRAY C90 system.
crayj90	True if you are on a CRAY J90 system.
crayts	True if you are on a CRAY T90 system.
crayct3d	True if you are on a CRAY T3D system.
crayct3e	True if you are on a CRAY T3E system.
crayxmp	True if you are on a CRAY Y-MP or a CRAY X-MP system.
crayymp	True if you are on a CRAY Y-MP system.
crayympe	True if you are on a system running IOS model E.
crayympel	True if you are on a CRAY Y-MP EL system.
pdp11	True if you are on a PDP-11/45 or PDP-11/70, always false.

`sparc`	True if you are on a computer using a SPARC-family processor, always false.
`sun`	True if you are on a Sun system, always false.
`u370`	True if you are on a UNIX/370 system, always false.
`u3b5`	True if you are on a 3B5 system, always false.
`u3b`	True if you are on a 3B20S, always false.
`vax`	True if you are on a VAX-11/750 or VAX-11/780, always false.

The commands that do not apply will return a false (nonzero) value. These commands are often used within `make`(1) makefiles and shell procedures to increase portability.

SEE ALSO

`make`(1), `sh`(1), `target`(1), `test`(1), `true`(1)

NAME

`mail` – Invokes an electronic message system

SYNOPSIS

```
mail [-e] [-f file]
mail [-e -p] [-q] [-r] [-f file]
mail [-t] persons
```

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The `mail` utility invokes an electronic mail system. It can be used to both send and receive mail; you can also use it, for example, from a script.

Receiving Mail

When you log in, the system notifies you if you have mail. It also tells you if new mail arrives while you are using `mail`.

The following options control the way in which received mail is read:

`-e` Prevents mail from being displayed. An exit value of 0 is returned if the user has mail; otherwise, an exit value of 1 is returned.

`-f` *file* Reads `mail` from *file* (such as `mbox`), rather than the default mail file (`/usr/mail/$LOGNAME`).

`-p` Displays all mail without prompting for disposition.

`-q` Terminates `mail` after interrupts. Usually, an interrupt causes the termination of only the message being displayed.

`-r` Displays messages in first-in, first-out order.

The `mail` command without arguments prints your mail, message by message, in last-in, first-out order. For each message, you are prompted with a `?`, and your response determines the disposition of the message. `mail` also recognizes the following commands when you are reading received mail:

`<newline>` Goes on to next message or stops if there are no more messages.

`+` Functions the same as a newline character.

`d` Deletes the message and goes on to the next message; it deletes only messages at your active security label.

`p`	Prints (displays) message again. It saves a message at your active security label, which may cause the message to be relabeled.
`-`	Returns to previous message, even if it was deleted.
`s` [*files*]	Saves the message, including its header, in the specified *files* (`mbox` is default). If the file to which the message is being saved already exists, it appends the message to it. It saves a message at your active security label, which may cause the message to be relabeled.
`w` [*files*]	Saves the message, without its header (the line that contains the sender's name and postmark), in the specified *files* (`mbox` is default). It saves a message at your active security label, which may cause the message to be relabeled.
`m` [*persons*]	Mails the message to the specified *persons* (default is to you). The message is labeled at your active security label when it is mailed.
`q`	Puts undeleted mail back in the mail file (`/usr/mail/$LOGNAME`) and stops. Mail messages that are not at your security label are not modified.
`<CONTROL-d>`	Functions the same as `q`.
`x`	Puts all mail, including deleted mail, back unchanged in the mail file (`/usr/mail/$LOGNAME`) and stops.
`!`*command*	Escapes to the shell to execute *command*.
`*`	Prints a command summary.

The following options control the way in which received mail is read:

`-e`	Prevents mail from being displayed. An exit value of 0 is returned if the user has mail; otherwise, an exit value of 1 is returned.
`-f` *file*	Reads `mail` from *file* (such as `mbox`), rather than the default mail file (`/usr/mail/$LOGNAME`).
`-p`	Displays all mail without prompting for disposition.
`-q`	Terminates `mail` after interrupts. Usually, an interrupt causes the termination of only the message being displayed.
`-r`	Displays messages in first-in, first-out order.

Sending Mail

The persons to whom you send mail (*persons*) are usually user names recognized by `login`(1). If a person being sent mail is not recognized, or if `mail` is interrupted during input, the mail is saved in the `$HOME/dead.letter` file, which can be edited and resent. The dead letter file is overwritten each time mail is mis-sent.

The following option and argument control the way in which mail is sent:

`-t`	Includes all *persons* to whom mail was sent in a line in each recipient's mail header.

persons When you specify *persons*, `mail` takes the standard input up to an end-of-file (or up to a line that consists of just a `.`) and adds it to each person's mail file (`/usr/mail/$LOGNAME`). The message is preceded by the sender's name and a postmark. Lines that look like postmarks in the message (that is, `From . . .`) are preceded with a `>`.

You can manipulate the mail file (`/usr/mail/$LOGNAME`) in two ways to alter the function of `mail`. The *other* permissions of the file may be read-write, read-only, or neither read nor write to allow different levels of privacy. If changed to other than the default, the file will be preserved even when empty to perpetuate the desired permissions. The file may also contain the first line:

```
Forward to
person
```

This causes all mail sent to the owner of the mail file to be forwarded to *person*. This is especially useful for forwarding all of a person's mail to one machine in a multiple-machine environment. For forwarding to work properly, the mail file should have `mail` as group ID and the group permission should be read-write.

NOTES

If your system is using multilevel directories (MLDs), you must define the directories that contain your system mailbox and your user saved mailbox as a MLD. Your security administrator should be responsible for defining MLDs.

If your system mailbox is not set up as an MLD, mail delivery to users at more than one security label will be disrupted.

If your saved mailbox directory is not defined as an MLD, you will be able to save mail messages only when you are logged in at the security label of your saved mailbox directory.

Depending on system configuration, you may not be allowed to see announcements of received mail at labels to which you do not have mandatory access control (MAC) read access. If you are allowed to receive announcements and you have mail at a label to which you do not have MAC read access, you will be informed that you have unreadable mail at a specific label.

If mail at a certain label is forwarded to another system, you will receive notification in the following form:

```
Your mail at <label> is being forwarded to <destination>.
```

This notification is given for mail at labels to which you have MAC read access.

Although `mail` allows you to save any mail message you receive, the saved version of the mail message is created at the label at which it was read, not necessarily the label at which it was sent.

Although you can forward or send any mail message, the message is always transmitted at your active label. If you decided to forward a message that has a lower security label than your active security label, the message is relabeled with your active security label and then forwarded.

The `mail` command uses `sendmail`(8) as the mechanism to send mail to a remote system.

EXIT STATUS

The `mail` utility exits with the following values:

0 Successful completion when the user had mail.

1 The user had no mail or an initialization error.

>1 An error occurred after initialization.

BUGS

Race conditions sometimes result in a failure to remove a lock file.

After an interrupt, the next message may not be printed; to force printing type a `p`.

A user's mailbox is labeled with the security label of the mail that is contained in the mailbox. Any attempt to send mail to a user at a different security label will fail, and mail can be lost.

EXAMPLES

The following command sends the mail message shown on the second line to users `joe` and `sam`:

```
mail joe sam
This should send mail to Joe and Sam.
.
```

The following command sends the file `memo` to user `sue`:

```
mail sue <memo
```

FILES

`/etc/udb`	User validation file that contains user control limits
`HOME/dead.letter`	Text that could not be mailed
`HOME/mbox`	Saved mail
`/tmp/ma*`	Temporary file
`/usr/mail/*.lock`	Lock for mail directory
`/usr/mail/`*user*	Incoming mail for *user* (the mail file)

SEE ALSO

`login`(1), `write`(1)

`chown`(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

`sendmail`(8) in the *UNICOS Administrator Commands Reference Manual*, Cray Research publication SR–2022

NAME

`mailq` – Prints the contents of the mail queue

SYNOPSIS

`mailq [-v]`

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `mailq` utility prints a summary of the mail messages queued for future delivery.

The first line printed for each message includes its internal identifier used on the host, its size (in bytes), the date and time it was accepted into the queue, and its envelope sender. The second line includes the error that caused the message to be retained in the queue; an error message will not be present if the message is being processed for the first time.

The `mailq` utility is identical to `sendmail -bp`.

The `mailq` command accepts the following flag:

`-v` Prints verbose information. The first line printed for each message will also include the message priority and an indicator ("+") if a warning message has been sent. Additional lines may be present, indicating the "controlling user." These indicate the owner of any programs that are executed on behalf of a message and the alias name (if any) that expanded the command.

The `mailq` utility exits with a value of 0, if successful, and >0 when an error occurs.

NOTES

If this utility is installed with a privilege assignment list (PAL), a user who is assigned the following privilege text upon execution of this command is allowed to perform the actions shown:

Privilege Text	Action
`daemon`	Allowed to see all information in the mail queue
`mailq`	Allowed to see all information in the mail queue

If this utility is installed with a PAL, a user with one of the following active categories is allowed to perform the action shown:

Active Category	Action
`system`, `secadm`	Allowed to see all information in the mail queue

If the PRIV_SU configuration option is enabled, the super user is allowed to see all information in the mail queue.

SEE ALSO

sendmail(8) in the *UNICOS Administrator Commands Reference Manual*, Cray Research publication SR–2022

NAME

`mailx` – Invokes an electronic message processing system

SYNOPSIS

Send Mode:

`mailx` [`-h` *number*] [`-r` *address*] [`-s` *subject*] [`-F`] *user...*

Receive Mode:

```
mailx -e
mailx [-d] [-H] [-i] [-n] [-N] [-T file] [-u user] [-U] [-V]
mailx -f [-d] [-H] [-i] [-I] [-n] [-N] [-T file] [-U] [-V] [file]
```

Obsolescent version; may not be supported in future releases:

`mailx` [`-f` [*file*]] [`-d`] [`-H`] [`-i`] [`-I`] [`-n`] [`-N`] [`-T` *file*] [`-U`] [`-V`]

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4
AT&T extensions (`-d`, `-I`, `-T`, `-U`, and `-V` options)

STANDARDS

POSIX, XPG4

DESCRIPTION

The `mailx` utility invokes an electronic mail system. You can use it to send and receive mail.

When you are reading mail, `mailx` lets you save, delete, and respond to messages. When you are sending mail, `mailx` lets you edit, review, and perform other modifications to the messages as you enter them.

The `mailx` utility stores incoming mail in a standard file (*mailbox*) for each user. When you call `mailx` to read messages, the *mailbox* is the default place to find them. As `mailx` reads messages, it marks them to be moved to a secondary file for storage, unless you specify that you want something else done with them. This secondary file is called the `mbox`, and it usually is located in the user's `HOME` directory (see `MBOX` in the ENVIRONMENT VARIABLES section for a description of this file). Messages remain in this file until you remove them.

You can access a secondary file by using the `-f` option of the `mailx` utility. Messages in the secondary file can then be read or otherwise processed using the same `mailx` commands as in the primary *mailbox*.

The operands that follow options are assumed to be destinations (or recipients). If you do not specify recipients, `mailx` attempts to read messages from the system *mailbox*.

The `mailx` utility accepts the following options:

`-d`	Turns on debugging output. This option is not recommended.
`-e`	Tests for presence of mail. If mail to read exists, `mailx` prints nothing and exits with a successful return code.
`-f` [*file*]	Reads messages from *file*, rather than *mailbox*. If you omit *file*, `mbox` is used.
`-F`	Records the message in a file named after the first recipient. If set (see the ENVIRONMENT VARIABLES section), this option overrides the `record` variable.
`-h` *number*	Specifies the number of network "hops" made so far. This option is provided for network software to avoid infinite delivery loops.
`-H`	Prints only header summary.
`-i`	Ignores interrupts. See also `ignore` in the Internal Variables subsection.
`-I`	Includes the newsgroup and article-ID header lines when printing mail messages. You must specify the `-f` option with this option.
`-n`	Prevents initialization from the system default `mailx.rc` file.
`-N`	Prevents printing of initial header summary.
`-r` *address*	Passes *address* to network delivery software.
`-s` *subject*	Sets the subject header field to *subject*.
`-T` *file*	Records message-ID and article-ID header lines in *file* after the message is read. This option also sets the `-I` option.
`-u` *user*	Reads the system mailbox that belongs to *user*. This is successful only if the invoking user has the appropriate privileges to read the system mailbox of that user.
`-U`	Converts uucp style addresses to Internet standards. Overrides the `conv` internal variable. Disables all tilde commands. This option is effective only if the system mailbox that belongs to *user* is not read protected.
`-V`	Prints the `mailx` version number and exits.
user...	Recipients of mail.
file	Read messages from *file*, rather than *mailbox*. If you omit *file*, `mbox` is used.

When mail is being read, `mailx` is in command mode. A header summary of the first several messages is displayed, followed by a prompt that indicates that `mailx` can accept regular commands (see the Commands subsection). When mail is being sent, `mailx` is in *input mode*. If you omit a subject on the command line, a prompt for the subject is printed. As you type the message, `mailx` reads the message and stores it in a temporary file. You can enter commands by beginning a line with the tilde (`~`) escape character, followed by one command letter and optional arguments. See the Tilde Escapes subsection for a summary of these commands.

At any time, `mailx`'s behavior is governed by a set of internal variables. These are flags and valued parameters that are set and cleared by using the `set` and `unset` commands. See the Internal Variables subsection for a summary of these parameters.

Recipients listed on the command line can be of three types: login names, shell commands, or alias groups. Login names can be any network address, including mixed-network addressing. If the recipient name begins with a pipe (`|`) symbol, the rest of the name is considered a shell command through which to pipe the message. This provides an automatic interface with any program that reads the standard input, such as `lp`(1), for recording on paper outgoing mail. Alias groups, set by the alias (`a`) command (see the Commands subsection) are lists of recipients of any type.

Regular commands are of the following form:

[*command*] [*msglist*] [*arguments*]

If you do not specify a command in command mode, the print command (`p`) is assumed. In input mode, commands are recognized by the escape character, and lines not treated as commands are considered input for the message.

Each message is assigned a sequential number, and when it is a current message, it is marked by a `>` in the header summary. Many commands take an optional list of messages (*msglist*) on which to operate, which defaults to the current message. A *msglist* is a list of message specifications separated by spaces that can include the following:

n	Message number *n*
`.`	Current message
`^`	First undeleted message
`$`	Last message
`*`	All messages
n-m	An inclusive range of message numbers
user	All messages from *user*
/string	All messages with *string* in the subject line (case ignored)

`:`*c* All messages of type *c*; *c* is one of the following:

- `d` Deleted messages
- `n` New messages
- `o` Old messages
- `r` Read messages
- `u` Unread messages

The context of the command determines whether this type of message specification makes sense.

Other arguments are usually arbitrary strings whose usage depends on the command involved. File names, when expected, are expanded using the typical shell conventions (see `sh`(1)). Special characters are recognized by certain commands, and are documented with the commands that follow.

At start-up time, `mailx` reads commands from a system-wide file (`/usr/lib/mailx/mailx.rc`) to initialize certain parameters, then from a private start-up file (`$HOME/.mailrc`) for personalized variables. Most regular commands are legal inside start-up files, the most common use being to set up initial display options and alias lists. The following commands are not legal in the start-up file: `!`, `Copy`, `edit`, `followup`, `Followup`, `hold`, `mail`, `preserve`, `reply`, `Reply`, `shell`, and `visual`. Any errors in the start-up file cause the remaining lines in the file to be ignored. The `.mailrc` file is optional, and it must be constructed locally.

Commands

The following is a complete list of `mailx` commands:

`!`*command* Invokes the command interpreter specified by `SHELL`. If you set the bang (`!`) variable, each unescaped occurrence of `!` in *command* is replaced with the command executed by the previous `!` command or `~!` tilde escape. See `SHELL` in the ENVIRONMENT VARIABLES section.

 # comment Null command (comment). This might be useful in `.mailrc` files.

`=` Prints the current message number.

`?` Prints a summary of commands.

`a`[`lias`] *alias names* ...
`g`[`roup`] *alias names* ...
Declares an alias for the given names. When *alias* is used as a recipient, the *names* will be substituted. Useful in the `.mailrc` file.

`alt`[`ernates`] *names* ...
Declares a list of alternative names for your login. When responding to a message, these names are removed from the list of recipients for the response. Without arguments, `alternates` prints the current list of alternative names. See also `allnet` in the Internal Variables subsection.

`cd` [*directory*]
`ch`[`dir`] [*directory*] Changes directory. If you omit *directory*, `$HOME` is used.

c[opy] [*file*]
c[opy] [*msglist*] *file* Copies messages to the file without marking the messages as saved; otherwise, it is equivalent to the `save` command. If a message is labeled at a lower security label, the copied message is relabeled with your active security label.

C[opy] [*msglist*] Saves the specified messages in a file whose name is derived from the author of the message to be saved, without marking the messages as saved; otherwise, it is equivalent to the `save` command. If a message is labeled at a lower security label, the copied message is relabeled with your active security label.

d[elete] [*msglist*] Deletes messages from the *mailbox*. If you set `autoprint`, the next message after the last one deleted is printed (see the Internal Variables subsection). It deletes messages only at your active security label.

di[scard] [*header-field* ...]
ig[nore] [*header-field* ...]
Suppresses printing of the specified header fields when displaying messages on the screen. Examples of header fields to ignore are `status` and `cc`. The fields are included when the message is saved. The `Print` and `Type` commands override this command.

dp [*msglist*]
dt [*msglist*] Deletes the specified messages from the *mailbox* and prints the next message after the last one deleted. Roughly equivalent to a `delete` command, followed by a `print` command. Deletes messages only at your active security label.

ec[ho] *string* ... Echoes the given strings (such as `echo`(1)).

e[dit] [*msglist*] Edits the given messages. The messages are placed in a temporary file, and the `EDITOR` environment variable is used to get the name of the editor (see the ENVIRONMENT VARIABLES section). Default editor is `ed`(1). Editing of a message is done at your current security label. If the message cannot be written back to the mail file at that label, the edited message is discarded when the mail box is closed.

ex[it]
x[it] Exits from `mailx`, without changing the *mailbox*. No messages are saved in the `mbox` (see also `quit`).

fi[le] [*file*]
fold[er] [*file*] Quits from the current file of messages and reads in the specified file. Several special characters are recognized when used as file names, with the following substitutions:

%	The current *mailbox*
%*user*	The *mailbox* for *user*
#	The previous file
&	The current *mbox*

+*file* The named file in the `folder` directory (see the `folder` variable)

The default file is the current *mailbox*.

`folders` Prints the names of the files in the directory set by the `folder` variable (see the Internal Variables subsection).

`fo`[`llowup`] [*message*]

Responds to a message, recording the response in a file whose name is derived from the author of the message. Overrides the `record` variable if set. See also the `Followup`, `Save`, and `Copy` commands and `outfolder` in the Internal Variables subsection.

`F`[`ollowup`] [*msglist*] Responds to the first message in the *msglist*, sending the message to the author of each message in the *msglist*. The subject line is taken from the first message and the response is recorded in a file whose name is derived from the author of the first message. See also the `followup`, `Save`, and `Copy` commands, and `outfolder` in the Internal Variables subsection.

`f`[`rom`] [*msglist*] Prints the header summary for the specified messages.

`g`[`roup`] *alias names* . . .
`a`[`lias`] *alias names* . . .

Declares an alias for the given *names*. When *alias* is used as a recipient, the *names* will be substituted. Useful in the `.mailrc` file.

`h`[`eaders`] [*message*] Prints the page of headers that includes the specified message. The `screen` variable sets the number of headers per page (see the Internal Variables subsection). See also the `z` command.

`hel`[`p`] Prints a summary of commands.

`ho`[`ld`] [*msglist*]
`pre`[`serve`] [*msglist*] Holds the specified messages in the *mailbox*.

`i`[`f`] *s*|*r*
`""`*mail-command*s
`el`[`se`]
`""`*mail-command*s
`en`[`dif`] Specifies conditional execution; *s* executes the following *mail-command*s, up to an `else` or `endif`, if the program is in send mode, and *r* executes the *mail-command*s only in receive mode. Useful in the `.mailrc` file.

`ig`[`nore`] *header-field* . . .
`di`[`scard`] *header-field* . . .

Suppresses printing of the specified header fields when displaying messages on the screen. Examples of header fields to ignore are `status` and `cc`. When the message is saved, all fields are included. The `Print` and `Type` commands override this command.

`l`[`ist`]	Prints all commands available. No explanation is given.
`m`[`ail`] *name* ...	Mails a message to the specified users. It labels the mailed message at your active security label.
`M`[`ail`] *name*	Mails a message to the specified user and records a copy of it in a file with the same name as the user.
`mb`[`ox`] [*msglist*]	Arranges for the specified messages to be in the standard `mbox` save file when `mailx` terminates normally. See `MBOX` in the ENVIRONMENT VARIABLES section for a description of this file. See also the `exit` and `quit` commands.
`n`[`ext`] [*message*]	Goes to next message that matches *message*. A *msglist* can be specified; however, the first valid message in the list is the only one used. This is useful for jumping to the next message from a specific user, because the name would be taken as a command in the absence of a real command. See the previous discussion of *msglist*s for a description of possible message specifications.
`pi`[`pe`] [[*msglist*] *command*]	
`|` [[*msglist*] *command*]	Pipes the message through the given *command*. The message is treated as if it were read. Without arguments, the current message is piped through the command specified by the value of the `cmd` variable. If the `page` variable is set, it inserts a `<form-feed>` character after each message (see the Internal Variables subsection).
`pre`[`serve`] [*msglist*]	
`ho`[`ld`] [*msglist*]	Preserves the specified messages in the *mailbox*.
`P`[`rint`] [*msglist*]	
`T`[`ype`] [*msglist*]	Prints the specified messages on the screen, including all header fields. Overrides suppression of fields by the `ignore` command.
`p`[`rint`] [*msglist*]	
`t`[`ype`] [*msglist*]	Prints the specified messages. If you set `crt`, messages longer than the number of lines specified by the `crt` variable are paged through the command specified by the `PAGER` environment variable. The default command is `pg`(1) in the ENVIRONMENT VARIABLES section.
`q`[`uit`]	Exits from `mailx`, storing messages that were read in `mbox` and unread messages in the *mailbox*. Deletes messages that have been explicitly saved in a file. Messages that are saved in `mbox` are relabeled, if necessary, at your active security label.
`R`[`eply`] [*msglist*]	
`R`[`espond`] [*msglist*]	Sends a response to the author of each message in the *msglist*. The subject line is taken from the first message. If you set `record` to a file name, the response is saved at the end of that file (see the Internal Variables subsection). Any response to a message is labeled at your active security label, even if the reply is to a message at a lower security label.

r[eply] [*message*]
r[espond] [*message*] Replies to the specified message, including all other recipients of the message. If you set `record` to a file name, the response is saved at the end of that file (see the Internal Variables subsection).

S[ave] [*msglist*] Saves the specified messages in a file whose name is derived from the author of the first message. The name of the file is taken to be the author's name with all network addressing stripped off. See also the `Copy`, `followup`, and `Followup` commands, and `outfolder` in the Internal Variables subsection. Messages can be saved only at your active security label; if a message with a lower security label is saved, it is relabeled with your active security label.

s[ave] [*file*]
s[ave] [*msglist*] *file* Saves the specified messages in the given file. The file is created if it does not exist, or the message is appended to it if it does exist. The message is deleted from the *mailbox* when `mailx` terminates, unless `keepsave` is set (see in the Internal Variables subsection and the `exit` and `quit` commands). Messages can be saved only at your active security label; if a message with a lower security label is saved, it is relabeled with your active security label.

se[t]
se[t] *name*
se[t] *name=string*
se[t] *name=number* Defines a variable called *name*. The variable can be given a null, string, or numeric value. `set` by itself prints all defined variables and their values. See the Internal Variables subsection for detailed descriptions of the `mailx` variables.

sh[ell] Invokes an interactive shell (see `SHELL` in the ENVIRONMENT VARIABLES section).

si[ze] [*msglist*] Prints the size of the specified messages in number of characters

so[urce] *file* Reads commands from the specified file and returns to command mode.

to[p] [*msglist*] Prints the top few lines of the specified messages. If you set the `toplines` variable, it is taken as the number of lines to print (see in the Internal Variables subsection). The default is 5.

tou[ch] [*msglist*] Touches the specified messages. If any message in *msglist* is not specifically saved in a file, it will be placed in the `mbox` on normal termination. See `exit` and `quit`.

T[ype] [*msglist*]
P[rint] [*msglist*] Prints the specified messages on the screen, including all header fields. Overrides suppression of fields by the `ignore` command.

t[ype] [*msglist*]

p[rint] [*msglist*] Prints the specified messages. If you set `crt`, messages that are longer than the number of lines specified by the `crt` variable are paged through the command specified by the `PAGER` environment variable. The default command is `pg`(1) in the ENVIRONMENT VARIABLES section.

u[ndelete] [*msglist*] Restores the specified deleted messages. Will restore only messages that were deleted in the current mail session. If you set `autoprint`, the last message of those restored is printed (see the Internal Variables subsection).

undi[scard]
unig[nore] Removes the specified header fields from the list being ignored.

uns[et] *name* ... Erases the specified variables. If the variable was imported from the execution environment (that is, a shell variable), it cannot be erased.

ve[rsion] Prints the current version and release date.

v[isual] [*msglist*] Edits the specified messages by using a screen editor. The messages are placed in a temporary file and the `VISUAL` environment variable is used to get the name of the editor (see the ENVIRONMENT VARIABLES section). Editing of a message is done at your current security label. If the message cannot be written back to the mail file at that label, the edited message is discarded when the mail box is closed.

w[rite] [*msglist*] *file* Writes the given messages on the specified file, minus the header and trailing blank line; otherwise, it is equivalent to the `save` command. Messages can be saved only at your active security label. If a message with a lower security label is saved, it is relabeled with your active security label.

x[it]
ex[it] Exits from `mailx` without changing the *mailbox*. No messages are saved in the `mbox` (see also `quit`).

z[+|-] Scrolls the header display forward or backward one screenful. The `screen` variable sets the number of headers displayed (see the Internal Variables subsection).

Tilde Escapes

You can enter the following commands only from input mode, by beginning a line with the tilde escape character (~). See `escape` (see the Internal Variables subsection) for changing this special character.

~! *command* Escapes to the shell.

~. Simulates end of file (terminates message input). **Warning:** Users using `rlogin` to connect to Cray Research systems should not use the tilde-dot (~.) character sequence to simulate end-of-file. The character sequence tilde-dot (~.) will be interpreted by `rlogin` first to disconnect the user from the Cray Research system. Use `<CTRL-d>` to terminate message input.

~: *mail-command*
~_ *mail-command* Performs the command-level request. Valid only when sending a message while reading mail.

~?	Prints a summary of tilde escapes.
~A	Inserts the autograph string.
~a	Inserts the autograph string `sign` into the message (see the Internal Variables subsection).
~b *name* ...	Adds the *name* to the blind carbon copy (`Bcc`) list.
~c *name* ...	Adds the *name* to the carbon copy (`Cc`) list.
~d	Reads in the `dead.letter` file. See `DEAD` in the Internal Variables subsection for a description of this file.
~e	Invokes the editor on the partial message. See also `EDITOR` in the ENVIRONMENT VARIABLES section.
~f [*msglist*]	Forwards the specified messages. Inserts the messages into the message, without alteration.
~h	Prompts for `Subject line` and `To`, `Cc`, and `Bcc` lists. If the field is displayed with an initial value, you can edit it as if you had just typed it.
~i *string*	Inserts the value of the specified variable into the text of the message (for example, ~A is equivalent to `~i Sign`).
~m [*msglist*]	Inserts the specified messages into the letter, shifting the new text to the right one tab stop. Valid only when sending a message while reading mail.
~p	Prints the message being entered.
~q	Quits from input mode by simulating an interrupt. If the body of the message is not null, the partial message is saved in `dead.letter`. See `DEAD` in the ENVIRONMENT VARIABLES section for a description of this file.
~r *file* ~< *file* ~< !*command*	Reads in the specified file. If the argument begins with an exclamation point (!), the rest of the string is taken as an arbitrary shell command and is executed, with the standard output inserted into the message.
~s *string* ...	Sets the subject line to *string*.
~t *name* ...	Adds the given *name* to the list.
~v	Invokes a preferred screen editor on the partial message. See also the `VISUAL` environment variable in the ENVIRONMENT VARIABLES section.
~w *file*	Writes the partial message onto the specified file, without the header.
~x	Exits as with ~q, except the message is not saved in `dead.letter`.
~| *command*	Pipes the body of the message through the specified *command*. If the *command* returns a successful exit status, the output of the command replaces the message.

Internal Variables

The following variables are internal `mailx` variables. You can set each internal variable by using the `mailx set` command at any time. To erase variables use the `unset` and `set no`*name* commands.

`allnet` — Treats all network names whose last component (login name) match as identical. This causes the *msglist* message specifications to behave similarly. Default is `noallnet`. See also the `alternates` command and the `metoo` variable.

`append` — On termination, appends messages to the end of the `mbox` file instead of prepending them. Default is `noappend`.

`ask`
`asksub` — Prompts for the subject if it is not specified on the command line by using the `-s` option. Enabled by default.

`askbcc` — Prompts for the `Bcc` list after the subject is entered. Default is `noaskcc`.

`askcc` — Prompts for the `Cc` list after the subject is entered. Default is `noaskcc`.

`autoprint` — Enables automatic printing of messages after `delete` and `undelete` commands. Default is `noautoprint`.

`bang` — Enables the special-casing of exclamation points (`!`) in shell escape command lines as in `vi`(1). Default is `nobang`.

`cmd`=*command* — Sets the default command for the `pipe` command. No default value.

`conv`=*conversion* — Converts `uucp` addresses to the specified address style. The only valid conversion is *internet*, which requires a mail delivery program that conforms to the RFC 822 standard for electronic mail addressing. By default, conversion is disabled. See also `sendmail`(8) and the `-U` command-line option.

`crt`=*number* — Pipes messages that have more than *number* lines through the command specified by the value of the `PAGER` environment variable (`pg`(1)). Disabled by default.

`debug` — Enables verbose diagnostics for debugging. Messages are not delivered. Default is `nodebug`.

`dot` — Reads a period on a line by itself during input from a terminal as end-of-file. Default is `nodot`.

`escape`=*c* — Substitutes *c* for the ~ escape character.

`flipr` — Reverses the meanings of the `R` and `r` commands. The default is `noflipr`.

`folder`=*directory* — Saves standard mail files. Expands user-specified file names that begin with a plus (`+`) by preceding the file name with this directory name to obtain the real file name. If *directory* does not start with a slash (`/`), `$HOME` is prepended to it. To use the plus (`+`) construct on a `mailx` command line, `folder` must be an exported `sh` environment variable. No default exists for the `folder` variable. See also `outfolder` in this subsection.

`header` Enables printing of the header summary when entering `mailx`. Enabled by default.

`hold` Preserves all messages that are read in the *mailbox* instead of putting them in the standard `mbox` save file. Default is `nohold`.

`ignore` Ignores interrupts while entering messages. Handy for noisy dial-up lines. Default is `noignore`.

`ignoreeof` Ignores end-of-file during message input. Input must be terminated by a period (.) on a line by itself or by the `~.` command. Default is `noignoreeof`. See also the `dot` variable in this subsection.

`indentprefix`=*string*
A string to be prefixed to each line that is inserted into the message by the `~m` command escape. The default for this variable is one tab character.

`keep` When the *mailbox* is empty, truncates it to a length of 0 instead of removing it. Disabled by default.

`keepsave` Keeps messages that have been saved in other files in the *mailbox* instead of deleting them. Default is `nokeepsave`.

`metoo` If your login appears as a recipient, prevents its deletion from the list. Default is `nometoo`.

`onehop` When responding to a message that was originally sent to several recipients, the other recipient addresses are usually forced to be relative to the originating author's machine for the response. This flag disables alteration of the recipients' addresses, improving efficiency in a network where all machines can send directly to all other machines (that is, one hop away).

`outfolder` Locates the files used to record outgoing messages in the directory specified by the `folder` variable, unless the path name is absolute. Default is `nooutfolder`. See `folder` in this subsection and the `Save`, `Copy`, `followup`, and `Followup` commands.

`page` Inserts a `<form-feed>` after each message sent through the pipe, when used with the pipe command. Default is `nopage`.

`prompt`=*string* Sets the *command mode* prompt to *string*. Default is `?`.

`quiet` Refrains from printing the opening message and version when entering `mailx`. Default is `noquiet`.

`record`=*file* Records all outgoing mail in *file*. Disabled by default. See also `outfolder` in this subsection.

`save` Enables saving of messages in `dead.letter` on interrupt or delivery error. See the `DEAD` environment variable in the ENVIRONMENT VARIABLES section for a description of this file. Enabled by default.

`screen`=*number* Sets the number of lines in a screenful of headers for the `headers` command.

`sendmail`=*command*	Alternate command for delivering messages. Default is `/usr/lib/sendmail`.
`sendwait`	Waits for background mailer to finish before returning. Default is `nosendwait`.
`showto`	When displaying the header summary and the message is from you, prints the recipient's name instead of the author's name.
`sign`=*string*	Identifies the variable inserted into the text of a message when the `~a` (autograph) command is given. No default (see also `~i` in the Tilde Escapes subsection).
`Sign`=*string*	Identifies the variable inserted into the text of a message when the `~A` command is specified. No default (see also `~i` in the Tilde Escapes subsection).
`toplines`=*number*	Indicates the number of lines of header to print with the `top` command. Default is 5.

NOTES

If your saved mail box directory is not defined as a multilevel directory (MLD), you can save mail messages only when logged in at the security label of your saved mail box directory.

Depending on system configuration, you may not be allowed to see announcements of received mail at labels to which you do not have MAC read access. If you are allowed to receive announcements and you have mail at a label to which you do not have MAC read access, you will be informed that you have unreadable mail at a specific label.

If mail at a certain label is forwarded to another system, you will receive notification in the following form:

```
Your mail at <label> is being forwarded to <destination>.
```

This notification is given for mail at labels to which you have MAC read access.

Although `mailx` allows you to save any mail message you receive, the saved version of the mail message is created at the label at which it was read, not necessarily the label at which it was sent.

Although you can forward or send any mail message, the message is transmitted at your active label. If you decide to forward a message that has a lower label than your active label, the message is relabeled with your active label and then forwarded.

ENVIRONMENT VARIABLES

Following are the environment variables for `mailx`:

`DEAD`	The path name of the file in which to save partial letters if untimely interrupts or delivery errors. Default is `$HOME/dead.letter`.
`EDITOR`	The command to run when the `edit` or `~e` command is used. Default is `ed`(1).
`HOME`	The user's base of operations; the user's home directory.
`LISTER`	A string that represents the command for writing the contents of the `folder` directory to standard output when the `folders` command is specified. The default is `ls`(1).

MAILRC The path name of the start-up file. Default is $HOME/.mailrc.

MBOX The path name of the file to save messages that have been read. The exit command overrides this function, as does saving the message explicitly in another file. Default is $HOME/mbox.

PAGER A string that represents an output filtering and/or pagination command for writing the output to the terminal. When standard output is a terminal device, the message output is piped through the command if the mailx internal variable crt is set to a value less than the number of lines in the message. Default is pg(1).

SHELL Identifies the name of a preferred command interpreter. Default is sh(1).

TERM If the internal variable screen is not specified, identifies the name of the terminal type to determine the number of lines in a screenful of headers.

VISUAL The name of a preferred screen editor. When the visual command or ~v command-escape is used, this editor will be invoked. Default is vi(1).

EXIT STATUS

The -e option is specified, mailx exits with one of the following values:

0 Mail was found.

>0 Mail was not found or an error occurred.

Otherwise, mailx exits with one of the following values:

0 Successful completion.

>0 An error occurred.

BUGS

You can use the -h and -r options only if mailx is using a delivery system program other than /usr/bin/rmail.

When *command* is shown as valid, arguments are not always allowed. Experimentation is recommended.

You cannot unset Internal variables imported from the execution environment.

The mailx utility does not fully support the full Internet addressing.

The sendmail(8) utility (the default mail delivery program) interprets a message that is a line consisting only of a ‘‘.’’ as the end of the message.

FILES

$HOME/.mailrc	Personal start-up file
$HOME/mbox	Secondary storage file
/tmp/R[emqsx]*	Temporary files

`/usr/lib/mailx/mailx.help*`	Help message files
`/usr/lib/mailx/mailx.rc`	Global start-up file
`/usr/mail/*`	Post office directory

SEE ALSO

`ls`(1), `mail`(1), `pg`(1), `sh`(1)

`sendmail`(8) in the *UNICOS Administrator Commands Reference Manual*, Cray Research publication SR–2022

NAME

make – Maintains, updates, and regenerates groups of programs

SYNOPSIS

make [-k] [-d] [-e] [-f *makefile*]... [-i] [-l] [-n] [-o] [-p] [-q] [-r] [-s] [-t] [*macro=name*]... [*target_name...*]

make [-S] [-d] [-e] [-f *makefile*]... [-i] [-l] [-n] [-o] [-p] [-q] [-r] [-s] [-t] [*macro=name*]... [*target_name...*]

IMPLEMENTATION

Cray PVP systems

STANDARDS

POSIX, XPG4
AT&T extensions (-d option)

DESCRIPTION

The make utility executes commands in *makefile* to update one or more target *target_name*s, which are typically programs.

The make utility examines time relationships and updates those derived files (called targets) that have modified times earlier than the modified times of the files (called prerequisites) from which they are derived. A description file (makefile) contains a description of the relationships between files, and the commands that must be executed to update the targets to reflect changes in their prerequisites. Each specification, or rule, consists of a target, optional prerequisites, and optional commands to be executed when a prerequisite is newer than the target. There are two types of rules:

- Inference rules, which have one target name with at least one period (.) and no slash (/)
- Target rules, which can have more than one target name

In addition, make has a collection of built-in macros and inference rules that infer prerequisite relationships to simplify maintenance of programs.

The system administrator may define a file named /etc/MAKEFILE, which can contain an additional set of macros, special targets, and inference rules that modify the built-in rules or add more definitions to them. This file enables a site to define a specific make environment valid for the entire site. The /etc/MAKEFILE file is automatically read in when make is invoked and before any user makefile is processed, unless the -l option is used.

When the -f option is not present, makefile, Makefile, s.makefile, and s.Makefile are tried in order. If *makefile* is -, standard input will be taken. More than one -f *makefile* argument pair may appear.

The make utility updates a target only if it depends on files that are newer than the target. All prerequisite files of a target are added recursively to the list of targets. Missing files are considered out-of-date.

After make has ensured that all of the prerequisites of a target are up-to-date, and if the target is out-of-date, the commands associated with the target entry are executed. If there are no commands listed for the target, the target is treated as up-to-date.

The following options are supported:

-d Writes to standard error detailed information on files and times examined.

-e Causes environment variables to override macro assignments within makefiles.

-f *makefile* Specifies a different makefile. The argument *makefile* is a pathname of a description file, also referred to as the *makefile*. A file name of - denotes standard input. There can be multiple occurrences of this option, and they are processed in the order specified. The contents of *makefile* override the built-in rules if they are present.

-i Ignores error codes returned by invoked commands. This mode is the same as if the special target name .IGNORE appears in the description file.

-k Continues to update other targets that do not depend on the current target if a non-ignored error occurs while executing the commands to bring a target up-to-date.

-l Ignores the file /etc/MAKEFILE and does not read it in when make is invoked.

-n Writes commands to standard output, but does not execute them. Lines with a plus-sign (+) prefix are executed. Lines with an at-sign (@) prefix are written to standard output.

-o Prints the contents of the /etc/MAKEFILE file (if it exists).

-p Writes to standard output the complete set of macro definitions and target descriptions.

-q Returns a zero exit value if the target file is up-to-date; otherwise, returns an exit value of 1. Targets are not updated. A command-line (associated with the targets) with a plus-sign (+) prefix is executed.

-r Clears the suffix list and does not use the built-in rules.

-s Does not write command lines or touch messages (see -t) before executing. This mode is the same as entered if the special target name .SILENT appears in the description file.

-S Terminates make if an error occurs while executing the commands to bring a target up-to-date. This is the default and the opposite of -k.

-t Updates the modification time of each target as though a touch *target* had been executed (see touch(1)). Targets that have prerequisites but no commands, or that are already up-to-date, are not touched in this manner. Write messages to standard output for each target file indicating the name of the file and that it was touched. Normally, the command lines associated with each target are not executed. However, a command line with a plus-sign (+) prefix is executed.

If the -k and -S options are both specified on the command line, by the MAKEFLAGS environment variable or by the MAKEFLAGS macro, the last one evaluated takes precedence. The MAKEFLAGS environment variable is evaluated first and the command line is evaluated second.

The following operands are supported:

macro=name Macro definitions. See the Macros subsection.

target_name Target names. If no target is specified, while make is processing the makefiles, the first target that make encounters that is not a special target or an inference rule is used.

Makefile Syntax

A makefile can contain rules, macro definitions, and comments. There are two kinds of rules: inference rules and target rules. The make utility contains a set of built-in inference rules. If the -r option is present, the built-in rules are not used and the suffix list is cleared. Additional rules of both types can be specified in a makefile. If a rules or macro is defined more than once, the value of the rule or macro is that of the last one specified. Comments start with a # symbol and continue until an unescaped newline character is reached.

The rules in makefiles consist of the following types of lines: target rules (including special targets), inference rules, macro definitions, empty lines, and comments.

When an escaped newline character (one preceded by a \ symbol) is found anywhere in the makefile, it is replaced, along with any leading white space on the following line, with a single space.

Makefile Execution

Command lines can have one or more of the following prefixes: a dash (-), an at sign (@), or a plus sign (+). These modify the way in which make processes the command. When a command is written to standard output, the prefix is not included in the output.

- If the command prefix contains a dash, or the -i option is present, or the special target .IGNORE has either the current target as a prerequisite or has no prerequisites, any error found while executing the command is ignored.

@ If the command prefix contains an at sign and the command-line -n option is not specified, or the -s option is present, or the special target .SILENT has either the current target as a prerequisite or has no prerequisites, the command is not written to standard output before it is executed.

\+ If the command prefix contains a plus sign, this indicates a command line is executed even if -n, -q, or -t is specified.

The -n option specifies printing without execution; however, if the command line contains the string $(MAKE), the line will be executed regardless of the -n option (see the discussion of the MAKEFLAGS macro under the Environment subsection). The -t (touch) option updates the date of target files without executing any commands.

Commands returning nonzero status normally terminate make. The error is ignored when the -i option is present, .IGNORE: appears in the makefile, or the initial character sequence of the command contains a dash. When the -k option is present, work stops on the current entry but continues on other branches that do not depend on that entry.

An interrupt, software termination, hangup, or quit signal received during the execution of a command line causes the associated target to be deleted, unless the target is a prerequisite of the special target `.PRECIOUS`. Text following a semicolon (`;`) character and all following lines that begin with a `<tab>` are shell commands to be executed to update the target.

The first nonempty line that does not begin with a `<tab>` or # begins a new prerequisite or macro definition. Shell commands may be continued across lines with the backslash character followed by a newline character. Everything printed by `make` (except the initial `<tab>`) is passed directly to the shell as is.

Command lines are executed one at a time, each by its own shell. The environment for the command being executed contains all of the variables in the environment of `make`. The macros from the command line to `make` are added to `make`'s environment. If any command-line macro has been defined elsewhere, the command-line value overwrites the existing value. By default, when `make` receives a nonzero exit status from the execution of a command, it terminates with an error message to standard error.

The following example makefile indicates that `pgm` depends on two files, `a.o` and `b.o`, and that they in turn depend on their corresponding source files, `a.c` and `b.c`, and a common file, `incl.h`:

```
pgm:    a.o b.o
        cc a.o b.o -o pgm
a.o:    incl.h a.c
        cc -c a.c
b.o:    incl.h b.c
        cc -c b.c
```

Target Rules

Target rules are formatted as follows:

target[*target*...]: [*prerequisite*...][;*command*]
[<tab>*command*
<tab>*command*
...]
(*line that does not begin with* <tab>)

Target entries are specified by a blank-separated, nonnull list of targets, followed by a colon, and then a blank-separated, possibly empty, list of prerequisites. Text following a semicolon, if any, and all following lines that begin with a `<tab>` are command lines to be executed to update the target. The first nonempty line that does not begin with a `<tab>` or # begins a new entry. An empty or blank line, or a line beginning with #, may begin a new entry.

Applications may select target names from the set of characters consisting solely of periods, underscores, digits, and alphabetics.

The following are recognized as special `make` targets:

`.DEFAULT` Uses the commands associated with the name `.DEFAULT` if it exists; these commands are used when a file must be made but there are no explicit commands or relevant built-in rules.

`.IGNORE` Prerequisites of this special target are targets themselves; this causes errors from commands associated with them to be ignored in the same manner as specified by the `-i` option. Subsequent occurrences of `.IGNORE` are added to the list of targets ignoring command errors. If no prerequisites are specified, `make` behaves as if the `-i` option had been specified and errors from all commands associated with all targets are ignored.

`.POSIX` This target is specified without prerequisites or commands. If it appears on a line by itself anywhere in any of the makefiles specified, `make` processes all makefiles in a manner defined by POSIX 1003.2. See the Suffixes subsection.

`.PRECIOUS` Prerequisites of this special target are not removed if `make` receives one of the following signals: quit, hang up, interrupt, and software termination. Subsequent occurrences of `.PRECIOUS` are added to the list of precious files. If no prerequisites are specified, all targets in the makefile are treated as if specified with `.PRECIOUS`.

`.SILENT` Prerequisites of this special target are targets themselves; this causes commands associated with them to not be written to the standard output before they are executed. Subsequent occurrences of `.SILENT` are added to the list of targets with silent commands. If no prerequisites are specified, `make` behaves as if the `-s` option had been specified and no commands or touch messages associated with any target is written to standard output.

`.SUFFIXES` Prerequisites of `.SUFFIXES` are appended to the list of known suffixes and are used in conjunction with the inference rules (see the Inference Rules subsection). If `.SUFFIXES` does not have any prerequisites, the list of known suffixes are cleared. Makefiles do not associate commands with `.SUFFIXES`.

Macros

Entries of the form *string1=string2* are macro definitions. The macro named *string1* is defined as having the value of *string2*, where *string2* is defined as all characters, if any, after the equal sign(`=`), up to a comment character (`#`) or an unescaped newline character. Any blanks immediately before or after the equal sign are ignored.

Subsequent appearances of `$(`*string1*`)` or `$(`*string2*`)` are replaced by *string2*. The parentheses or braces are optional if *string1* is a single character. The macro `$$` is replaced by the single character `$`.

Macro names are made up from the set of characters consisting solely of periods, underscores, digits, and alphabetics from the portable character set (that is, ASCII). A macro name may not contain an equal sign.

Macros can appear anywhere in the makefile. Macros in target lines are evaluated when the target line is read. Macros in command lines are evaluated when the command is executed. Macros in macro definition lines are not evaluated until the new macro being defined is used in a rule or command. A macro that has not been defined evaluates to a null string without causing an error condition.

The forms $(*string1*[:*subst1*=[*subst2*]]) or ${*string1*[:*subst1*=[*subst2*]]} can be used to replace all occurrences of *subst1* with *subst2* when the macro substitution is performed. The *subst1* to be replaced is recognized when it is a suffix at the end of a word in *string1* (a word in this context is defined to be a string delimited by the beginning of the line, a blank, or a newline character).

Macro assignments are accepted from the sources listed below, in the order shown. If a macro name already exists at the time it is being processed, the newer definition replaces the existing definition:

1. Macros defined in make's built-in inference rules.
2. The contents of the environment, including the variables with null values, in the order defined in the environment.
3. Macros defined in the makefile(s), processed in the order specified.
4. Macros specified on the command line.

If the -e option is specified, the order of processing sources 2 and 3 are reversed.

The VPATH macro, consisting of a list of colon-separated directory paths, may be used to specify the location of prerequisite files.

The SHELL macro is treated specially. It is provided by make and set to the path name of the shell command language interpreter, /bin/sh. The SHELL environment variable does not affect the value of the SHELL macro. If SHELL is defined in the makefile or is specified on the command line, it replaces the original value of the SHELL macro, but does not affect the SHELL environment variable.

The MAKEFLAGS environment variable, when processed by make, is assumed to contain any legal input option (except -f, -p, and -r) defined for the command line. Further, upon invocation, make creates the variable if it is not in the environment, puts the current options into it, and passes it on to invocations of commands. Thus, MAKEFLAGS always contains the current input options. This proves very useful for *super-makes*, which are master makefiles that call other makefiles (these are used for system builds). The MAKEFLAGS environment variable may also contain option letters without the leading dashes (-) and no separating blanks.

In fact, as noted previously, when the -n option is used, the $(MAKE) command is executed anyway; hence, you can specify make -n recursively on a whole software system to see what would have been executed. This is because the -n is put in MAKEFLAGS and passed to further invocations of $(MAKE). This is one way of debugging all makefiles for a software project without actually executing anything.

Inference Rules

Inference rules can be made shorter, as in this example:

```
pgm:    a.o b.o
        cc a.o b.o -o pgm
a.o b.o:        incl.h
```

The shorter form is possible because make has a set of internal rules for building files. A user may add rules to this list by simply putting them in the makefile.

Certain macros are used by the default inference rules to permit the inclusion of optional matter in any resulting commands. For example, CFLAGS and YFLAGS are used for compiler options to cc(1) and yacc(1), respectively. The previously stated method for examining the current rules is recommended.

The inference of prerequisites can be controlled. The rule for creating a file with suffix .o from a file with suffix .c is specified as an entry with .c.o: as the target and no prerequisites. Shell commands associated with the target define the rule for making a .o file from a .c file. Any target that has no slashes in it and starts with a dot is identified as a rule and not a true target.

Libraries

If a target or prerequisite name contains parentheses, it is assumed to be an archive library, the string within parentheses referring to a member in the library. Thus, lib(file.o) and $(LIB)(file.o) both refer to an archive library that contains file.o (assuming that the LIB macro has been previously defined). The expression $(LIB)(file1.o file2.o) is not legal. Rules pertaining to archive libraries have the form *.XX*.a, in which the *XX* is the suffix from which the archive member is to be made. An unfortunate byproduct of the current implementation requires that *XX* be different from the suffix of the archive member. Thus, lib(file.o) cannot depend on file.o explicitly.

The most common use of the archive interface follows; it assumes that the source files consist of all C-type source code:

```
lib: lib(file1.o) lib(file2.o) lib(file3.o)
        @echo lib is now up-to-date
c.a:
        $(CC) -c $(CFLAGS) $<
        $(AR) $(ARFLAGS) $@ $*.o
        rm -f $*.o
```

The .c.a: rule listed previously is built into make and is not necessary in this example. A more interesting but more limited example of an archive library maintenance construction follows:

```
lib:    lib(file1.o) lib(file2.o) lib(file3.o)
        $(CC) -c $(CFLAGS) $(?:.o=.c)
        $(AR) $(ARFLAGS) lib $?
        rm $?
        @echo lib is now up-to-date
c.a:;
```

In this example, the substitution mode of the macro expansions is used. The $? list is defined as the set of object file names (inside lib) whose C source files are out-of-date. The substitution mode translates .o to .c. Note also the disabling of the .c.a: rule, which would have created each object file, one by one. This particular construct speeds archive library maintenance considerably. This type of construct becomes very cumbersome if the archive library contains a mix of assembly programs and C programs.

Internal Macros

The following macros are maintained internally and are useful in writing rules for building targets:

`$*` The `$*` macro represents the file name part of the current target name with the suffix deleted. It is evaluated only for inference rules.

For example, in the `.c.a` inference rule, `$*.o` represents the out-of-date `.o` file that corresponds to the prerequisite `.c` file.

`$@` The `$@` macro represents the full target name of the current target. It is evaluated only for explicitly named dependencies.

For example, in the `.c.a` inference rule, `$@` represents the out-of-date `.a` file to be built. Similarly, in a makefile target rule to build `lib.a` from `file.c`, `$@` represents the out-of-date `lib.a`.

`$<` The `$<` macro is evaluated only for inference rules or the `.DEFAULT` rule. It is the module that is out-of-date with respect to the target (that is, the ''manufactured'' dependent file name). Thus, in the `.c.o` rule, the `$<` macro would evaluate to the `.c` file. Following are two examples for making `.o` files from `.c` files:

```
.c.o:
        cc -c $.c
```

or:

```
.c.o:
        cc -c $<
```

`$?` The `$?` macro is evaluated when explicit rules from the makefile are evaluated. It is the list of prerequisites that are out-of-date with respect to the target (essentially, the modules that must be rebuilt).

For example, in a makefile target rule to build `prog` from `file1.o`, `file2.o`, and `file3.o`, and where `prog` is not out of date with respect to `file1.o`, but is out of date with respect to `file2.o` and `file3.o`, `$?` represents `file2.o` and `file3.o`.

`$%` The `$%` macro is evaluated only when the current target is an archive library member of the form *libname*`(`*member*`.o)`. In these cases, `$@` evaluates to *libname* and `$%` evaluates to *member*`.o`. The `$%` macro is evaluated for both target and inference rules.

For example, in a makefile target rule to build `lib.a(file.o)`, `$%` represents `file.o`, as opposed to `$@`, which represents `lib.a`.

Each of the internal macros can have alternative forms. When an uppercase `D` or `F` is appended to any of these macros, the meaning is changed to *directory part* for `D` and *file part* for `F`. The directory part is the path prefix of the file without a trailing slash; for the current directory, the directory part is a period (`.`). When the `$?` macro contains more than one prerequisite file name, the `$(?D)` and `$(?F)` (or `${?D}` and `${?F}`) macros expand to a list of directory name parts and file name parts respectively.

For the target *lib*(*member*.o) and the .*s2*.a rule, the internal macros are defined as follows:

$< *member*.*s2*

$* *member*

$@ *lib*

$? *member*.*s2*

$% *member*.o

Include Files

If the string include or sinclude appears at the beginning of a line in a makefile and is followed by a <blank> or a <tab>, the rest of the line is assumed to be a file name and will be read by the current invocation of make after substituting for any macros. If the file is not readable, it is a fatal error for include, and silently ignored by sinclude. After reading the include file, the remainder of the makefile will be read. Include files can be nested up to 16 levels.

Suffixes

Certain names (for instance, those ending with .o) have prerequisites, such as .c or .s, that can be inferred. If no update commands for such a file appear in *makefile*, and if a prerequisite that can be inferred exists, that prerequisite will be compiled to make the target. In this case, make has inference rules that allow files to be built from other files when the suffixes are processed and the appropriate inference rule is used.

When the special target .POSIX is used, the following rules are defined:

```
.c .c.a .c.o
.f .f.a .f.o
.l.c .l.o
.sh
.y.c .y.o
```

When .POSIX is not used, all of the preceding rules, in addition to the ones following, are defined:

```
.c~.a .c~.c .c~.o .C .C~ .C.o .C~.C .C~.o .C.a .C~.a
.f .f~ .f~.f .f.a .f~.a .f.o .f~.o
.F .F~ .F~.F .F.a .F~.a .F.o .F~.o
.f90 .f90~ .f90~.f90 .f90.a .f90~.a .f90.o .f90~.o
.F90 .F90~ .F90~.F90 .F90.a .F90~.a .F90.o .F90~.o
.h~.h
.l~.c .l~.l .l~.o
.p .p~ .p.a .p~.a .p.o .p~.o .p~.p
.s.a .s~.a .s.o .s~.o .s~.s
.sh~ .sh~.sh
.y~.c .y~.o .y~.y
```

The internal rules for the make utility can be printed in a form suitable for modifications and for inclusion in a user's makefiles. To print the make default built-in rules, use the following standard shell command:

```
make -fpl - 2>/dev/null </dev/null
```

To print the make default built-in rules as modified by the existence of the /etc/MAKEFILE file, use the following standard shell command:

```
make -fp - 2>/dev/null </dev/null
```

Note: If the /etc/MAKEFILE file contains the special target .POSIX, this command will list make default built-in POSIX rules as modified by the contents of /etc/MAKEFILE.

To print only the contents of /etc/MAKEFILE, use the following standard shell command:

```
make -fo - 2>/dev/null </dev/null
```

To print the make built-in POSIX rules, create a file named makefile containing the special target .POSIX and execute the following standard shell command:

```
make -pl 2>/dev/null
```

To print the built-in POSIX rules as modified by the contents of /etc/MAKEFILE, create a file named makefile containing the special target .POSIX and execute the following shell command:

```
make -p 2</dev/null
```

Note: If /etc/MAKEFILE already contains the special target .POSIX, it is not necessary to create the makefile containing this target.

The only peculiarity in this output is the (null) string that printf(3C) prints when handed a null string.

A tilde (~) in the above rules refers to a Source Code Control System (SCCS) file. Thus, the rule .c~.o would transform an SCCS C source file into an object file (.o). Because the s. of the SCCS files is a prefix, it is incompatible with the make suffix point of view. Hence, the tilde is a way of changing any file reference into an SCCS file reference.

A rule with only one suffix (that is, .c:) is the definition of how to build *x* from *x*.c. In effect, the other suffix is null. This is useful for building targets from only one source file (such as shell procedures and simple C programs).

Additional suffixes are specified as the prerequisite list for .SUFFIXES. Order is significant; the first possible name for which both a file and a rule exist is inferred as a prerequisite. The suffix list when the special target .POSIX is used is as follows:

```
.SUFFIXES: .o .c .y .l .a .sh .f
```

When .POSIX is not used, the list is as follows:

```
.SUFFIXES: .o .c .c~ .C .C~ .f .f~ .F .F~ .f90 .f90~ .F90 .F90~ .p .p~ .y .y~ .l .l~ .s .s~ .sh .sh~ .h .h~
```

The command for printing the internal rules (shown earlier in this subsection) displays the list of suffixes implemented on the current machine. Multiple suffix lists accumulate; `.SUFFIXES:` without prerequisites clears the list of suffixes.

VPATH Macro

When `make` searches for targets and prerequisites, it expects to find them in the current (`.`) directory or in the directory specified by their file names. For example, in the following rule, `make` would look for the file `file.c` in the `.` directory:

```
file.o: file.c
```

The `VPATH` macro allows you to specify alternate paths for the file search. The variable consists of the directory names separated by a colon (`:`), in the style of the `PATH` environmental variable.

The following example specifies a path containing three directories: `newsrc`, `src`, and `obj`.

```
VPATH=newsrc:src:obj
```

The `make` utility will search the directories in the following order: the current (`.`) directory, followed by `newsrc`, `src`, and `obj` for all dependencies (prerequisites) and targets. The first match will stop the search for a given file, even if the same file exists in the directories that might still follow in the `VPATH` and are newer versions of the file than the one found first. This allows you to write `make` rules as if all files existed in the current directory.

With the example `VPATH=newsrc:src:obj`, the same rule (`file.o: file.c`) would be interpreted as if it was written like this:

```
obj/file.o: src/file.c
```

This example interpretation works only if `file.o` and `file.c` do not exist in the current directory, but are found in the `obj` and `src` directories, respectively, and `file.c` does not exist in the `newsrc` directory and `file.o` does not exist in the `newsrc` or `src` directories.

Use caution when writing `make` rules to allow `make` to look for dependencies and targets in alternate directories using the `VPATH` variable. Typically, there is nothing wrong with the following rule:

```
VPATH=dir
file.o: file.c
      $(CC) $(CFLAGS) -c file.c
```

when the file `file.c` is found in the current directory. The target `file.o` will be created in the current directory.

However, if `file.c` is found in the `dir` directory instead of the current directory, this example would be asking `make` to execute the following rule:

```
file.o: dir/file.c
      $(CC) $(CFLAGS) -c dir/file.c
```

Although make can adapt the dependency line (file.o: file.c) to the VPATH requirements, it cannot edit the command line. The command lines have to execute as written.

The make internal variables ($*, $@, $<, $?, $%) allow you to fully use the VPATH macro. These variables are correctly prefixed with the directory path coming from the VPATH search (if applicable). The following rule will execute correctly in all cases:

```
VPATH=dir
file.o: file.c
      $(CC) $(CFLAGS) -c -o $@ $<
```

For example, if both file.o and file.c are not found in the current directory, but exist in the dir directory, the internal variables will be set to the following, and the command line will be executed correctly:

```
$@ = dir/file.o
$< = dir/file.c
```

To take full advantage of the VPATH macro, many implicit make rules have been augmented by adding the output file option to the compilation line, instead of using the compiler default output file. The new method is compatible with older code. The following characteristics are maintained for implicit rules:

- After execution of the make command, only the input (prerequisites) and the target files remain.
- Should execution of an implicit rule require generation of temporary files, such as when executing the SCCS command, lex, yacc, and so on, the temporary files are always created in the current (.) directory and removed at the end of the action. Thus, write permission for the user to the . directory is required for successful execution of make in these cases.

To examine one possible use of the VPATH macro, assume that all project sources are stored in the src directory and all object files (relocatables, binaries, and libraries) are stored in the obj directory. Three object directories have been created: ./obj_debug, ./obj_normal, and ./obj_optim. These directories contain binaries for the debug version, the standard version, and the final, highly optimized version, respectively.

You could set three aliases (the following example is in the ksh shell), setting yourself CFLAGS as appropriate:

```
alias maked='make CFLAGS=... VPATH=src:obj_debug'
alias maken='make CFLAGS=... VPATH=src:obj_normal'
alias makeo='make CFLAGS=... VPATH=src:obj_optim'
```

You would then place the makefile in the . directory and, by executing the appropriate alias, generate the version corresponding to the immediate needs. maked would generate a version suitable for use with the debugger and would place the binaries in the corresponding directory. Likewise, maken would generate the normal version in its own directory. All this would be done from the same sources without any change of the makefile or the current directory.

Alternately, you could copy several sources out of the src directory, modify them, and execute the same aliases. According to the VPATH search rules, modified versions in the . directory will be located before the versions in the src directory and will be used to generate the appropriate binary versions. When the modified sources become stable, they can be moved back to the src directory, where they are the new base for the development process. In this manner it is easy to store the same sources for different binaries of the same program for different purposes.

This example succeeds because make will place the target file on the same place where it found it using the VPATH search, or in the . directory if it didn't find it anywhere. Consequently, the first execution of make in the program will cause all targets (usually binaries) to be put into the . directory because they do not yet exist anywhere. The targets must be moved into the appropriate obj_... directory by hand. The process described above is consistent from then on.

The same is true if, during the program development process, you add several new source files. The corresponding binaries must be moved to the appropriate obj_... directory by hand after the first binaries were generated into the . directory.

The VPATH macro does not currently work with POSIX rules.

Multiprocessing

Compilations may be multiprocessed when the environment variable NPROC is set to a value greater than 1. NPROC determines the maximum number of processes to be run in the background. You can set NPROC to any value, but extremely large values can overload the system. If you set NPROC in the makefile, the value you specify will take precedence over other NPROC settings unless the -e option is used. Set NPROC=1 in a makefile to inhibit parallel processing.

To force synchronization, include a slash (/) in a prerequisite list. All targets preceding a slash in a prerequisite list are completed before anything following the slash begins. The make command line is not multiprocessed.

If a slash (/) is used to force multiprocessing and the root file system has a date newer than that of the target, the target is executed.

NOTES

File names with the characters =, :, and @ do not work.

Commands executed directly by the shell, notably cd(1), are ineffectual across newline characters in make.

You cannot build lib(file.o) from file.o.

EXIT STATUS

When the -q option is specified, the make utility exits with one of the following values:

0 Successful completion.

1 The target was not up-to-date.

>1 An error occurred.

When the -q option is not specified, the make utility exits with one of the following values:

0 Successful completion.

>0 An error occurred.

FILES

/etc/MAKEFILE	File containing site default rules, read in before any user makefile
makefile	First file searched
Makefile	Second file searched
s.makefile	Third file searched
s.Makefile	Fourth file searched

SEE ALSO

ar(1), get(1), cd(1), lex(1), sh(1), yacc(1)

cc(1) in the *Cray Standard C Reference Manual*, Cray Research publication SR–2074

printf(3C) in the *UNICOS System Libraries Reference Manual*, Cray Research publication SR–2080

Managing Projects with make, Talbott, Steve and Oram, Andrew, O'Reilly & Associates, Inc., 1991.

NAME

makekey – Generates encryption key

SYNOPSIS

/usr/lib/makekey

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The makekey utility improves the usefulness of encryption schemes that depend on a key by increasing the amount of time required to search the key space. It reads 10 bytes from its standard input and writes 13 bytes on its standard output. The output depends on the input in a way intended to be difficult to compute.

The first 8 input bytes (the *input key*) can be arbitrary ASCII characters. The last 2 characters (the *salt*) are best chosen from the set of digits, ., /, uppercase letters, and lowercase letters. The salt characters are repeated as the first 2 characters of the output. The remaining 11 output characters are chosen from the same set as the salt and constitute the *output key*.

The transformation performed is essentially the following: the salt is used to select one of 4096 cryptographic machines, which are all based on the National Bureau of Standards DES algorithm, but broken in 4096 various ways. Using the *input key* as key, a constant string is fed into the machine and recirculated several times. The 64 bits that come out are distributed into the 66 *output key* bits in the result.

The makekey utility is intended for programs that perform encryption (such as ed(1), vi(1), and crypt(1)). Usually, its input and output are pipes.

NOTES

Inclusion of the Data Encryption Standard (DES) encryption code requires a special license for sites outside the United States and Canada. If these encryption functions are unavailable on your system, check with your system administrator or site analyst.

SEE ALSO

crypt(1), ed(1), vi(1)

NAME

man – Displays or prints online man page information

SYNOPSIS

man [-l] [-M *path*] [-q] [-t] [-s *section*] [-T *macro*] *title* ...
man [-M *path*] -f *filename* ...
man [-M *path*] -i *keyword* ...
man [-M *path*] -k *pattern* ...

IMPLEMENTATION

UNICOS systems

STANDARDS

POSIX, XPG4
BSD extensions (-f, -M, -t, and -T)
CRI extensions (-i and -q)

DESCRIPTION

The man utility displays online man pages that you select by title, or it can display entries about a particular topic by using the -f, -i, and -k options.

The man utility accepts the following options:

-l — Lists all pages found matching the *title*(s) given the search path. The search path is a colon-separated list of directories in which man page subdirectories may be found.

-M *path* — Changes the directory path that man searches for man page information (default is /usr/man). The search path is a colon-separated list of directories in which man page subdirectories may be found (for example, /usr/local:/usr/man). When you use this option with the -f, -i, or -k option, the -M option must appear first.

-q — Displays the quick-reference entry for the specified command manual entry. Currently, quick-reference entries are provided only for man pages in sections 1, 2, 3, and 8.

-s *section* ... — Specifies sections of the manual for man to search. The directories searched for *title*(s) is limited to those specified by *section*(s). *section* is an Arabic number or one of the words 'new', 'local', 'old', 'public', or 'quick'. A section may be followed by a one-letter classified. To specify multiple sections, separate each section with a comma.

-t — Processes the specified man page entries through troff(1), with the output piped to lpr -n. Use of the -t option on a system where the unformatted source versions of the man pages are not available causes an error.

`-T` *macro* Specifies a different macro package to use when processing `nroff` source versions of man pages. By default, the Cray Research man page macros (`-muc`), defined in `/usr/lib/tmac/tmac.uc`, are used. The format of the macro option argument can be either a full path name or the actual option to be used in the `nroff` or `troff` command line (for example, `-man` and `/usr/lib/tmac/tmac.an` are both valid and are equivalent).

`-f` *filename* ... Displays from the `whatis` file (table of contents), one-line summaries that contain any of the specified file names. The leading path name components are stripped from each file name before the search is performed.

`-i` *keyword* ... Displays from the `index` file, formatted, three-line summaries that contain any of the specified keywords.

`-k` *pattern* ... Displays from the `whatis` file (table of contents), one-line summaries that contain the specified pattern. Grep-style pattern matching characters may be used in *pattern* (see `grep`(1)).

title ... Pipes output through `cat`(1).

section Specifies the section number of manual to be searched.

When you specify *section* and *title* (*section* being an Arabic number, such as `3`, or one of the words `new`, `local`, `old`, or `public`), `man` searches that section of the manual for the specified title. A section number may be followed by a one-letter classifier (for example, `1b`, indicating a utility originating from BSD source code in section 1). If you omit *section*, `man` will search all sections of the manual.

If the standard output is not a terminal, or if you specify the `-` flag, `man` will pipe its output through `cat`(1). Otherwise, `man` pipes its output through `more -s` to handle paging and underlining on the screen (see `more`(1)).

If both the `nroff`(1) source code and preformatted pages are available, the one with the most recent modification time will be displayed. If the source is used and if the user has appropriate permission, the newly formatted page will be installed in the directory of preformatted pages so that it will not have to be formatted the next time it is accessed.

NOTES

The `man` utility displays bold text by overstriking the normal font. If the bold font is specified, it must be the same height and width as the normal font.

The usage of '`man` *section title*' becomes obsolescent, which should be removed from future releases.

ENVIRONMENT VARIABLES

The `man` utility uses the following environment variables:

`MANPATH` When set, its value is used as the search path for man page searching. The `-M` option overrides this variable.

PAGER	When set, its value is used as the utility for displaying the requested man pages. By default, `more -s` is used.
TCAT	When set, its value is the name of the utility used to print `troff` output. By default, `lpr -n` is used.
TROFFCMD	When set, its value is used as the format utility called by the `-t` option. By default, `troff` is used.

FILES

`/usr/man/man?/*`	Directories containing unformatted man pages.
`/usr/man/cat?/*`	Directories containing preformatted pages.
`/usr/lib/tmac/tmac.uc`	Cray Research man page macros.
`/usr/man/index`	File that contains cross-reference information on all UNICOS man page entries; used by the `-i` option.
`/usr/man/whatis`	File that contains all man page entry name lines; used by the `-f` and `-k` options.

SEE ALSO

`apropos`(1), `cat`(1), `col`(1), `grep`(1), `more`(1), `nroff`(1), `troff`(1), `whatis`(1)

NAME

mesg – Permits or denies messages

SYNOPSIS

```
mesg [y | n]
```

Obsolescent version: may not be supported in future releases:

```
mesg -y
mesg -n
```

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The mesg utility controls whether other users are allowed to send messages by using write(1), talk(1B), or other utilities to a terminal device. The terminal device affected is determined by searching for the first terminal in the sequence of devices associated with standard input, standard output, and standard error, respectively. With no operands, mesg reports the current state without changing it.

Processes with appropriate privileges may be able to send messages to the terminal device independent of its current state.

The mesg utility accepts the following operands:

y Grants permission to other users to send messages to the terminal device.

n Denies permission to other users to send messages to the terminal device.

The obsolescent version accepts the following options:

-y Equivalent to the y operand.

-n Equivalent to the n operand.

NOTES

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
system, secadm	Allowed to bypass all protections on the terminal device.

If the PRIV_SU configuration option is enabled, the super user is allowed to override all protections on the terminal device.

EXIT STATUS

The mesg utility exits with one of the following values:

0 Receiving messages is allowed.

1 Receiving messages is not allowed.

>1 An error occurred.

EXAMPLES

The following example first determines your current message setting and then changes it. User input is shown in bold font:

```
$ mesg
is y
$ mesg n
$ mesg
is n
```

FILES

/dev/tty*	Device files

SEE ALSO

write(1), talk(1B)

General UNICOS System Administration, Cray Research publication SG–2301

NAME

mkdir – Creates a directory

SYNOPSIS

mkdir [-L *level*[,*compartment*[,*compartment*[,...]]] [-m *mode*] [-p] *directories*

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4
CRI extensions (-L option)

DESCRIPTION

The mkdir utility creates specified directories in mode 777 (unless altered by umask(1)). Standard entries . (for the directory itself) and .. (for its parent) are made automatically. mkdir cannot create these entries by name.

The mkdir utility requires write permission in the parent directory.

The owner ID of the new directory is set to the process' real user ID. The group ID of the new directory is set to the group ID of the parent directory.

The mkdir utility accepts the following options:

-L *level*[,*compartment*[,*compartment*[,...]]
: Allows the directory to be relabeled to the specified security label. Choices for *level* and *compartments* are defined by your site security administrator. When the -L option is specified with the -p option, only the last directory in the path is labeled at the requested label. All intermediate directories are created at the user's active label. The behavior of a mkdir request requires that your active security label must be the same as the security label of the parent directory. Also, if the -L option is specified, the directory is relabeled to the specified security label. If the relabeling fails, the directory's label remains at your active label. The requested label must dominate your active label.

-m *mode*
: Lets you specify the *mode* to be used for new directories. You can find choices for modes in chmod(1). In the *symbolic_mode* strings, the op characters + and - are interpreted relative to an assumed initial mode of a=rwx; + adds permissions to the default mode, - deletes permissions from the default mode.

-p
: Creates a directory by creating all nonexisting parent directories first.

directories
: Specifies the name of the directory you are creating.

`mkdir` can fail for the following reasons:

- Your security level is not within the upper and lower security levels and/or authorized compartments of the file system.
- The new directory does not conform to the security level hierarchy rules (that is, the security level of the new directory must be equal to or greater than the security level of the parent directory). In addition, the security compartments of the directory to be created must be a proper superset of the parent directory compartments.

NOTES

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
`system`, `secadm`	Allowed to create any directory.
`sysadm`	Allowed to create any directory, subject to security label restrictions. Shell-redirected I/O is subject to security label restrictions.

If the `PRIV_SU` configuration option is enabled, the super user is allowed to create any directory.

EXIT STATUS

The `mkdir` utility returns exit code 0 when all directories are successfully made; otherwise, it prints a diagnostic message and returns a nonzero exit code.

EXAMPLES

To create subdirectory structure `ltr/jd/jan`, enter the following:

```
mkdir -p ltr/jd/jn
```

SEE ALSO

`chmod`(1), `rm`(1), `sh`(1), `umask`(1)

General UNICOS System Administration, Cray Research publication SG–2301

NAME

`mkfifo` – Makes FIFO special files

SYNOPSIS

`mkfifo` [`-m` *mode*] *file* ...

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The `mkfifo` utility creates the first in, first out (FIFO) special files specified by the operands, in the order specified.

The `mkfifo` utility accepts the following options and operand:

`-m` *mode* Sets the file permission bits of the newly created FIFO to the specified *mode* value. The *mode* argument is the same as the *mode* operand defined for the `chmod`(1) utility. In the *symbolic_mode* strings, the *op* characters `+` and `-` are interpreted relative to an assumed initial mode of `a=rw`.

file Specifies a path name of the FIFO special file to be created.

NOTES

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
`system`, `secadm`	Allowed to create any file.
`sysadm`	Allowed to create any file, subject to security label restrictions on the file's path. Shell-redirected I/O is subject to security label restrictions.

If the `PRIV_SU` configuration option is enabled, the super user is allowed to create any file.

EXIT STATUS

The `mkfifo` utility exits with one of the following values:

0 All of the specified FIFO special files were created successfully.

>0 An error occurred.

SEE ALSO

`chmod`(1)

`mkfifo`(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

General UNICOS System Administration, Cray Research publication SG–2301

NAME

more, page – File perusal filter for CRT viewing

SYNOPSIS

more [-ceisu] [-n *number*] [- *linenum*] [- *lines*] [-/ *pattern*] [-p *command*] [-t *tagstring*] [*file...*]

Obsolescent versions; may not be supported in future releases:

more [-ceisu] [-n *number*] [+*command*] [-t *tagstring*] [*file...*]
more [-ceisu] [-n *number*] [+*number*] [-t *tagstring*] [*file...*]
more [-cdefilsu] [-n *number*] [-p *command*] [-t *tagstring*] [*file...*]
more [-ceisu] [-*number*] [-p *command*] [-t *tagstring*] [*file...*]

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The more utility is a filter that lets you examine continuous text one screenful at a time. It typically pauses after each screenful. If you then press <return>, one more line is displayed. If you press the space bar, another screenful is displayed. Other possibilities are listed later in this section.

The more utility accepts the following options:

-c Draws each page by beginning at the top of the screen and erasing each line just before drawing on it. This avoids scrolling the screen, making it easier to read while more is writing. If the terminal cannot clear to the end of a line, this option is ignored.

-d (Obsolescent) Prompts you with the Press space to continue, 'q' to quit message at the end of each screenful. This is useful when you are using more as a filter in a setting, such as a class, in which many users may be unsophisticated.

-e Exits immediately after writing the last line of the last file in the argument file list.

-f (Obsolescent) Counts logical, rather than physical screen lines (that is, it does not fold long lines).

-i Performs pattern matching in searches, ignoring case.

-l (Obsolescent) Does not treat the <form-feed> character specially. If you do not specify this option, more will pause after any line that contains a <form-feed> character, as if the end of a screenful was reached. Also, if a file begins with a <form-feed>, the screen will be cleared before the file is printed.

`-n` *number*
`-`*number* (Obsolescent)
Specifies the number of lines per screenful. *number* is a positive decimal integer. The `-n` option overrides any value obtained from the environment.

`-p` *command*
`+`*command* (Obsolescent)
For each file examined, initially executes the `more` utility in the *command* argument. If *command* is a positioning command, such as a line number or a regular expression search, the current position is set to represent the final results of the command, without writing any intermediate lines of the file.

`-s` Squeezes multiple blank lines from the output, producing only one blank line. This option maximizes the useful information present on the screen.

`-t` *tagstring* Writes the screenful of the file containing the tag named by the *tagstring* argument. If both the `-t` *tagstring* and `-p` *command* option are specified, the `-t` *tagstring* is processed first. The file containing the tag is selected by `-t`, and then the command is executed.

`-u` Suppresses underline processing. Usually, `more` handles underlining such as that produced by `nroff`(1) in a manner appropriate to the particular terminal: if the terminal can perform underlining or has a stand-out mode, `more` will output appropriate escape sequences to enable underlining or stand-out mode for underlined information in the source file.

`+`*linenum* Starts up at line *linenum*.

If you invoke `more` as `page`, the screen is cleared before each screenful is printed (but only if a full screenful is being printed), and $k - 1$ rather than $k - 2$ lines are printed in each screenful (k is the number of lines the terminal can display).

The `more` utility looks in the `/usr/lib/terminfo` directory to determine terminal characteristics and the default window size defined by `TERM` environment variable. On a terminal that can display 24 lines, the default window size is 23 lines.

The `more` utility looks in the `MORE` environment variable to preset any desired flags. For example, if you view files by using the `-c` mode of operation, the `sh`(1) utilities will cause all invocations of `more`, including invocations by other programs, to use this mode. Only flags that are preceded with a hyphen (`-`) are recognized when using the `MORE` environment variable.

```
MORE=-c
export MORE
```

When `more` is reading from a file rather than a pipe, a percentage is displayed along with the *file* prompt. This specifies the fraction of the file (in characters, not lines) that has been read so far.

You may type other sequences when `more` pauses, and their effects are as follows (*i* is an optional integer argument, defaulting to 1):

[*i*]`f`
[*i*]`<CONTROL-f>` Moves forward *i* lines; default is one screenful.

[*i*]b
[*i*]<CONTROL-b> Moves backward *i* lines; default is one screenful.

[*i*]<space>
[*i*]j
[*i*]<newline> Scrolls forward *i* lines. The default for <space> is one screenful; for j and <newline>, one line.

[*i*]k Scrolls backward *i* lines; default is 1.

[*i*]d
[*i*]<CONTROL-d> Scrolls forward *i* lines; default is one half of the screen size. If *i* specify, *i* becomes the new default for subsequent d and u commands.

[*i*]s Skips *i* lines and prints a screenful of lines.

[*i*]u
[*i*]<CONTROL-u> Scrolls backward *i* lines; default is one half of the screen size. If you specify *i*, it becomes the new default for subsequent d and u commands.

[*i*]g Goes to line *i* in the file; default is 1 (beginning of file).

[*i*]G Goes to line *i* in the file; default is the end of the file.

r
<CONTROL-L> Redraws screen.

R Refreshes the screen, discarding any buffered input.

m*letter* Marks the current position with the specified *letter*; *letter* represents the name of one of the lowercase letters of the character set.

'*letter* Returns to the position that was previously marked with *letter*.

'' Returns to the position from which the last large movement command was executed (a large movement is defined as any movement of more than a screenful of lines). If no such movements were made, returns to the beginning of the file.

[*i*]/[!]*pattern*<newline>
Searches forward in the file for the *i*th line that contains *pattern*. The *i* defaults to 1. If the ! character is included, the lines that do not contain *pattern* are not searched.

[*i*]?[!]*pattern*<newline>
Searches backward in the file for the *i*th line that contains *pattern*. The *i* defaults to 1. If you include the character !, the lines that do not contain *pattern* are not searched.

[*i*]n Repeats the previous search for the *i*th line (default is 1) that contains the last pattern (or not containing the last pattern, if the previous search was /! or ?!).

[*i*]N Repeats the search in the opposite direction of the previous search for the *i*th line (default 1) containing the last pattern (or does not contain the last pattern if the previous search was /! or ?!).

`:e` [*filename*]`<newline>`
: Examines a new file. If you omit *filename*, the current file from the list of files in the command line is reexamined.

[*i*]`:n`
: Examines the next file; if you specify *i*, the *i*th next file is examined.

[*i*]`:p`
: Examine the previous file; if you specify *i*, the *i*th previous file is examined.

`:t` *tagstring*`<newline>`
: Displays the file beginning with the line that contains *tagstring*.

`v`
: Invokes an editor to edit the current file being examined. The name of the editor is taken from the `EDITOR` environment variable or defaults to `vi`.

`=`
`<CONTROL-g>`
: Displays the current line number.

`h`
: Help command; provides a description of all `more` commands.

`q`
`:q`
`ZZ`
: Exits `more`.

The commands take effect immediately; that is, it is not necessary to press `<RETURN>`.

The terminal is set to `noecho` mode by this program so that the output can be continuous.

NOTES

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
`system`, `secadm`	In a privileged administrator shell environment, shell-redirected I/O is not subject to file protections.
`sysadm`	Shell-redirected output is subject to security label restrictions.

If the `PRIV_SU` configuration option is enabled, shell-redirected I/O on behalf of the super user is not subject to file protections.

EXIT STATUS

The `more` utility exits with one of the following values:

0 Successful completion.

>0 An error occurred.

EXAMPLES

The use of `more` in the previewing of `nroff`(1) output is as follows:

```
nroff -ms doc.n | more -s
```

FILES

`/usr/lib/terminfo` Terminal database

SEE ALSO

`cat`(1), `nroff`(1), `pg`(1), `sh`(1)

NAME

msgi – Sends informative message to operator

SYNOPSIS

msgi *msg_string*

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The msgi utility lets you send an informative message to the operator.

The msgi utility accepts the following operand:

msg_string Specifies the text of the message that you want to sent.

The operator cannot reply to informative messages. All messages are logged in the order in which they are received.

Messages are not issued to operators to let them know that there is outstanding information. To see the msgi messages, operators must run the infd(8) command within the oper(8) command.

NOTES

You must enclose shell special characters or strings that contain shell special characters in quotation marks (single or double); if these are not enclosed in quotation marks, the shell will misinterpret the special characters. For example, the following messages contain a shell special character, the semicolon:

```
msgi "Tape ABC is not on the system; waiting for mount."

msgi Tape ABC is not on the system";" waiting for mount.
```

EXAMPLES

The operator executes the following oper(8) command, which starts the infd(8) command:

```
oper infd
```

The user sends the following message:

```
msgi from Al: See Bob for tape XYZ
```

The operator sees the following message:

```
Command: infd  Page: 1 [delay 10] Fri Jun  4 09:39:38 1993

from Al: See Bob for tape XYZ
              .
              .
              .
> _
```

FILES

`/usr/spool/msg/msglog.log`	Log file

SEE ALSO

`msgr`(1)

`infd`(8), `msgd`(8), `msgdaemon`(8), `msgdstop`(8), `oper`(8), `rep`(8) in the *UNICOS Administrator Commands Reference Manual*, Cray Research publication SR–2022

Tape Subsystem User's Guide, Cray Research publication SG–2051

NAME

msgr – Sends action message to operator

SYNOPSIS

msgr *msg_string*

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The msgr utility lets you send an action message to the operator and receive a reply from the operator.

The msgr utility accepts the following operand:

msg_string Specifies the text of the message that you want to sent.

The sending process is suspended until a reply is received from the operator. The reply is then printed to standard output. If you interrupt the process by using <CONTROL-c> while waiting for a reply, the action message will be canceled. All messages are logged in the order in which they are received.

To see these messages, the operator must run the msgd(8) command within the oper(8) command. The operator responds with the rep(8) command.

NOTES

You must enclose shell special characters or strings that contain shell special characters in quotation marks (single or double); if these are not enclosed in quotation marks, the shell will misinterpret the special characters. For example, the following messages contain shell special characters, a semicolon and question mark:

```
msgr "Please mount tape ABC; OK?"

msgr Please mount tape ABC";" OK"?"
```

EXAMPLES

The operator executes the oper(8) command (which runs msgd(8) if no command argument is provided):

```
$ oper
```

The user sends the following message:

```
$ msgr Please mount tape ABC located on shelf 29
```

The operator sees the message and responds by using the rep(8) command:

```
Command: msgd  Page: 1 [delay 10] Fri Jun  4 09:43:48 1993

  Msg #   Time    System Messages
  =====   =====   ===============

   799    09:43   From us1:  Please mount tape ABC located on shelf 29
                     .
                     .
                     .
> rep 799 Tape ABC found and mounted
> _
```

The user receives the reply:

```
$ msgr Please mount tape ABC located on shelf 29
Tape ABC found and mounted
$ _
```

FILES

/usr/spool/msg/msglog.log	Log file

SEE ALSO

msgi(1)

infd(8), msgd(8), msgdaemon(8), msgdstop(8), oper(8), rep(8) in the *UNICOS Administrator Commands Reference Manual*, Cray Research publication SR–2022

Tape Subsystem User's Guide, Cray Research publication SG–2051

NAME

mt – Issues commands to a magnetic tape drive

SYNOPSIS

/usr/ucb/mt [-f *tapename*] *command* [*count*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The mt utility issues commands to a magnetic tape drive. If *tapename* is not specified, the TAPE environment variable is used; if TAPE does not exist, the utility terminates. *tapename* must reference a character-special tape device. By default, mt performs the requested operation once. Operations may be performed multiple times by specifying *count*.

The mt utility accepts the following option and arguments.

-f *tapename* Specifies the character-special tape device to be activated.

command Specifies the command to execute on the tape device. Only as many characters as are required to uniquely identify a command need to be specified. Valid commands are as follows:

bsf [*count*]	Skips back over *count* file marks; the default is 1.
eof, weof	Writes *count* end-of-file marks at the current position on the tape.
fsf [*count*]	Skips forward over *count* file marks; the default is 1.
offline, rewoffl	Rewinds the tape and places the tape unit offline (*count* is ignored).
rewind	Rewinds the tape (*count* is ignored).
status	Prints status information about the tape unit.

count Specifies the number of files to skip over or the number of end-of-file marks to write.

NOTES

The mt utility can be used only for tape drives that are not configured up (UP).

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
secadm, sysadm	Allowed to use this command.

If the PRIV_SU configuration option is enabled, the super user is allowed to use this command.

EXIT STATUS

The mt utility exits with one of the following values:

0 The operation was successful.

1 The command was unrecognized.

2 The operation failed.

FILES

/dev/tape/*device_name*
Tape device node

SEE ALSO

dd(1)

ioctl(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

environ(7) (available only online)

Tape Subsystem User's Guide, Cray Research publication SG–2051

General UNICOS System Administration, Cray Research publication SG–2301

NAME

`mtdump` – Examines unformatted dump of multitasking history buffer

SYNOPSIS

`mtdump` [`-f` *form*] [`-t` *tsks*] [`-d` *data*] [`-a` *act*] [`-T`] [`-L`] [`-E`] [`-B`] [`-C`] [`-U`] [`-i` *intr*] [`-V`] *file*

IMPLEMENTATION

Cray PVP systems

DESCRIPTION

The `mtdump` command examines the unformatted dump of the multitasking history buffer and generates reports according to the options used. The only required argument is *file*; if no other options are specified, the report generated is a summary of all information contained in the dump. The default is the same as specifying `-f totals`.

You can generate reports in five primary formats, which display all the information in the buffer; you can select specific buffer entries, which display only portions of the available information; or you can select any combination of these formats, using the appropriate options.

Generating the five primary reports is specified by the following `-f` options. You can specify one or more `-f` options, each of which is separated by commas (with no blank spaces) and listed in any order. For each report you request, you will receive a complete listing, in the *form* format, of the entire multitasking history buffer file. Every report requested rewinds the file and displays the data in the requested *form*.

`-f` *form* Displays the multitasking history trace buffer in the specified format. Valid *form* arguments are as follows:

- `totals` Displays a summary of all events in multitasking history trace buffer, including the total number of occurrences of each kind of event. This is the default.
- `chron` Displays a chronological listing of all entries in the file.
- `sync` Displays synchronization points, with a separate column for each of up to 16 user tasks. If an event in the file does not involve task synchronization, it is not listed. The listing is grouped by task identifier.
- `cpu` Displays logical CPU use, with a separate column for each of up to 16 logical CPUs. A listing is made of the activities in each logical CPU. When the state of a logical CPU changes, a symbol appears in that CPU's column on the report, reflecting the new status of the CPU. This listing is grouped by CPU (not task). A map is produced, assigning each logical CPU to a UNICOS process ID.
- `status` Displays the status of each of up to 16 user tasks in uniform time intervals. You can change this interval using the `-i` *intr* option. Listing is by task identifier. Listing by a uniform time interval allows you to see the multitasking execution by a snapshot in time.

The following options let you list selected groups of buffer entries. If these options are specified, `mtdump` searches for and displays only those multitasking history buffer entries that match the specified criteria. The requested information is displayed in chronological order, in a format similar to the `chron` *form* previously described. If you specify any of the following options, with no `-f` option, no primary report is displayed; a single report is generated containing the selected items. Selected options can be combined to narrow the criteria for search. Note, however, that if you make the selection process too narrow, it is possible that no records will be found that meet the desired criteria. An informative message will be issued if no records are found that match the specified criteria.

`-t` *tsks* Selects one or more internal task identifiers, in decimal, separated by commas, for which buffer entries should be listed; maximum of 10 task identifiers is allowed.

`-d` *data* Selects one or more action-dependent data values, in octal, separated by commas, to be searched for; maximum of 10 entries for all data values is allowed.

`-a` *act* Selects one or more action codes of buffer entries, in decimal, separated by commas, to be listed, with a maximum of 40 actions allowed. Valid action codes are 0 through 127 (decimal). The default is to list entries for all action codes unless one or more of the `-t`, `-d`, `-T`, `-L`, `-E`, `-B`, `-C`, `-U`, or `-V` options has been used.

The following options allow you to specify a range of action codes to be searched for and displayed. Using the following options is simpler than trying to specify many values with the `-a` option.

If you specify one or more of these options, and also select actions by number (using `-a` *act*), the effect is cumulative. For example, if you specify `-a 34`, and also specify the `-B` option (barriers), records with action codes 33 through 37 are displayed. If you select the `-C` and `-B` options, records with action codes 21 through 37 are displayed.

`-T` Lists actions involving tasks; these include task starts, completions, waits, and tests. (Action codes 0 through 5.)

`-L` Lists actions involving locks. (Action codes 6 through 12.)

`-E` Lists actions involving events. (Action codes 13 through 20.)

`-B` Lists actions involving barriers. (Action codes 33 through 37.)

`-C` Lists actions involving logical CPUs. (Action codes 21 through 32.)

`-U` Lists actions involving user codes. (Action codes 64 through 127.)

Other options are as follows:

`-i` *intr* Specifies the number of clock periods, in decimal, for each time interval to be used in the `status` format display. The default is 1,000,000. This option has no effect on other format displays.

`-V` Displays the current `mtdump` version number, and a brief copyright message.

file Specifies the file containing the unformatted dump of the multitasking history trace buffer. Specifying *file* is required.

SEE ALSO

BUFTUNE(3F) in the *UNICOS System Libraries Reference Manual*, Cray Research publication SR–2080

performance(7) (available only online)

Guide to Parallel Vector Applications, Cray Research publication SG–2182

NAME

mv – Moves files or a directory

SYNOPSIS

mv [-f] [-i] *files target*

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

Use the mv utility to do any of the following:

- Move (rename) one file
- Move one or more files from a directory to another existing directory
- Rename a directory
- Move a directory

To move one or more files, specify the current file name(s) (*files*) and the new name for the file (*target*). Do not use the same name for *files* and *target*. (Be careful when using shell metacharacters.) If *target* is not a directory, only one file may be specified before it; if it is a directory, more than one file may be specified. If *target* does not exist, mv creates a file named *target*. If *target* exists and is not a directory, its contents are overwritten. If *target* is a directory, the *files* are moved to that directory. If *target* is a link to another file with links, the other links will remain and *target* will become a new file.

If mv determines that the mode of *target* forbids writing, it will print the mode (see chmod(1)), prompt for a response, and read the standard input for one line. If the line begins with y, mv will execute if permissible; if not, the utility will exit. When the standard input is not a terminal, no messages concerning mode restrictions are given.

The mv utility accepts the following options and operands:

-f Moves the *files* without prompting even if it is writing over an existing *target*. This is the default if the standard input is not a terminal. Any previous occurrences of the -i option are ignored.

-i Prompts for confirmation whenever the move would overwrite an existing *target*.

files Files to be moved.

target Destination of the moved file. A y answer means that the move should proceed. Any other answer prevents mv from overwriting the *target*. Any previous occurrences of the -f option are ignored.

Specifying both `-f` and `-i` is not an error. The last option specified determines the behavior of `mv`.

To move files from a directory to another existing directory, supply the file name(s) (*files)* and the directory name (*target*).

To rename a directory, specify the current directory name (*files*) and a new directory name (*target*). The new directory name must be unique and the two directories must have the same file system root. A file system that is not locally mounted must have root write permission for the directory rename to succeed.

Files with set-user-ID (SUID) and set-group-ID (SGID) permissions have those permissions removed on a move when *files* and *target* reside on different file systems, unless you have `suidgid` permission.

NOTES

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
`system`, `secadm`	Allowed to move any file or directory.
`sysadm`	Allowed to move any file or directory, subject to security label restrictions. Shell-redirected I/O is subject to security label restrictions.

If the `PRIV_SU` configuration option is enabled, the super user is allowed to move any file or directory.

If *files* and *target* reside on different file systems, `mv` must copy the file and delete the original. In this case, any linking relationship with other files is lost.

If a user tries to move a restart file to a different file system, it is no longer marked as a restart file and cannot be restarted.

EXIT STATUS

The `mv` utility exits with one of the following values:

0 All input files were moved successfully.

>0 An error occurred.

EXAMPLES

Example 1: Use the following command line to move file `yesterday` to `today`:

```
mv yesterday today
```

Example 2: Use the following command line to move all files in your working directory that begin with letter `r` to subdirectory `movehere`:

```
mv r* movehere
```

Example 3: Use the following command line to rename directory `olddir` to `newdir` (`newdir` does not already exist):

```
mv olddir newdir
```

SEE ALSO

`chmod`(1), `cp`(1), `cpio`(1), `ln`(1), `rm`(1), `sh`(1)

`chmod`(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

General UNICOS System Administration, Cray Research publication SG–2301

NAME

mvf – Provides a multivolume tape filter

SYNOPSIS

/bin/mvf -i [-a] [-b *blocksize*] [-c *vol_count*] [-C *copy_size*] [-d *digits*] [-f *file_id*] [-l *label_option*] [-n *buffers*] [-O *offset*] [-t] [-T *count*] [-v] [-V *volser_Base*] *device_name* [:*vsn*[,*vsn*]... [*device_name* [:*vsn*[,vsn]]]

/bin/mvf -o [-b *blocksize*] [-c *vol_count*] [-d *digits*] [-f *file_id*] [-l *label_option*] [-n *buffers*] [-t] [-T *count*] [-v] [-V *volser_Base*] *device_name* [:*vsn*[,*vsn*]... [*device_name* [:*vsn*[,vsn]]]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The mvf command is a filter that enables you to use multiple tape volumes as though a program were reading or writing to a very large tape volume. It accepts input from standard input (stdin) for an output operation to tape and writes to standard output (stdout) for an input operation from tape.

You must select the I/O direction by using either the -i option for input from tape or -o option for output to tape and the tape device path name (*device_name*). You may also specify volume serial numbers (VSNs) (*vsn*) following the device name.

If you specify the -V and -d options to construct the VSNs from a base VSN and a field width, mvf takes the base VSN and increments the relevant width argument as each new volume is required.

For an input operation, reading continues until the volume count (-c option) is exhausted or until all of the VSNs have been read.

For an output operation, writing continues until an EOF is detected on input or until the volume count (-c option) is exhausted. The output operation is allowed to extend to one more volume than is specified in the volume count specification.

The mvf command accepts the following options, arguments, and operands:

-a	Accepts read errors from the tape device. If read errors occur, the data written to stdout is unpredictable.
-b *blocksize*	Specifies a block size in bytes. The default is 98304.
-c *vol_count*	Specifies a count of volumes to process. The default is 1.
-C *copy_size*	Specifies the amount of data to copy to stdout for an input operation.
-d *digits*	Specifies the size of the digit field. See the -V option.
-f *file_id*	Specifies a file ID to be associated with all volumes written with either IBM or ANSI standard labels.

`-i` Reads from tape and writes to `stdout`. You cannot specify the `-i` option with the `-o` option.

`-l` *label_option* Sets the type of label processing to use. *label_option* may be any of the following:

`sl` Specifies IBM standard labels.

`al` Specifies ANSI standard labels.

`blp` Specifies bypass label processing. `blp` skips any label processing and reads or writes nonlabeled tapes. The tape is terminated by the device during close processing and is positioned so that a new file can be added to it.

To unload the tape after `blp` use, you must use the `mt`(1B) command or physically unload the tape device.

If you omit the `-l` option, `mvf` reads the label to verify that a nonlabeled tape has been loaded.

If you load a tape and its tape type differs from the type you specified, `mvf` reformats the tape with the specified label type.

`-n` *buffers* Specifies a number of asynchronous I/O buffers to use for each tape device. The default is 6.

`-o` Reads from `stdin` and writes to tape. You cannot specify the `-i` option with the `-o` option.

`-O` *offset* Specifies an offset into the tape record to extract data for an input operation.

`-t` Displays timing information.

`-T` *count* Sets the count of internal trace buffers to use for debugging and automatically dumps them at exit time to `stderr`.

`-v` Displays portions of the tape labels and performs a limited set of label consistency operations on labeled tapes. The default is no label consistency checking.

`-V` *volser_Base* Specifies the base VSN for internally generated VSN lists. See the `-d` option.

device_name [`:`*vsn*[`,`*vsn*]] Specifies character-special device names and a list of VSNs. At least one device name is mandatory, followed optionally by a colon and a list of VSNs separated by commas. When you specify multiple tape devices, `mvf` processes data on one device until it reaches the end of the tape. At that time, the tape device is unloaded, and the operation is continued on the next device.

For operations that require multiple tapes, device switching saves the rewind unload time and consequently maximizes throughput.

Specifying multiple tape devices enables input or output to continue without having to wait for the current device to rewind and unload.

EXAMPLES

The following examples illustrate different uses of the `mvf` command.

Example 1: This example shows how to back up the `/v` file system onto a tape without a tape label:

```
cd /v find . -depth -print | cpio -o | mvf -o /dev/tape/cart59
```

Example 2: This example shows how to restore the `/v/xyz/test.f` file from a previous backup of the `/v` file system:

```
mvf -i /dev/tape/cart59 | cpio -i ./xyz/test.f
```

Example 3: The following example shows how to back up the `/u` file system. It uses four tape volumes and ANSI label processing:

```
cd /u find . -depth -print | cpio -o | \
mvf -ol al  /dev/tape/cart59:q11010,q11011,q11012,q11013 \
```

Example 4: This example provides an alternative way to back up `/u`. You use two tapes mounted on two devices; the second device overlaps its I/O with the unload of the first device.

```
cd /u find . -depth -print | cpio -o | \
mvf -o -l al /dev/tape/cart10:q11010,q11012 \
/dev/tape/cart59:q11011,q11013
```

The volume usage sequence for this example is as follows:

```
write q11010 on /dev/tape/cart10
unload q11010, write q11011 on /dev/tape/cart59
unload q11011, write q11012 on /dev/tape/cart10
unload q11012, write q11013 on /dev/tape/cart59
unload q11013
```

Example 5: This example performs a level 0 dump of `*/ptmp` to a set of volumes.

```
/etc/dump -t0 -f- /ptmp | mvf -ol sl  302:000600,000602,000604
```

It produces the following screen output:

```
mvf: Mount 000600 on 302 OUTPUT mode
dump (/ptmp to -): Date of this level 0 dump: Thu Apr  6 13:44:43 1995
dump (/ptmp to -): Dumping /ptmp
dump (/ptmp to -): to -
dump (/ptmp to -): mapping (Pass I) [regular files]
dump (/ptmp to -): mapping (Pass II) [directories]
dump (/ptmp to -): estimated 229048 sectors on 0.00 volume(s).

dump (/ptmp to -): dumping (Pass III) [directories]
mvf: Mount 000601 on 303 OUTPUT mode
dump (/ptmp to -): dumping (Pass IV) [regular files]
mvf: 6808 blocks out on 302
mvf: Mount 000602 on 302 when Rewind/Unload is Complete.
mvf: 6814 blocks out on 303
mvf: Mount 000603 on 303 when Rewind/Unload is Complete.
mvf: 6816 blocks out on 302
mvf: Mount 000604 on 302 when Rewind/Unload is Complete.
mvf: 6826 blocks out on 303
mvf: Mount 000605 on 303 when Rewind/Unload is Complete.
dump (/ptmp to -): un-mapping (Pass V) [changed files]
dump (/ptmp to -): dump has completed, 229048 blocks
mvf: 28631 blocks on 5 cartridges
```

In this case, the last volume, 000605, was requested, but was not actually used.

NOTES

The `mvf` command does not reject duplicated options; it uses the last valid option of a particular type.

FILES

`/dev/tape/`*device_name*	Tape device node

SEE ALSO

`mt`(1B)

`tpinit`(8) in the *UNICOS Administrator Commands Reference Manual*, Cray Research publication SR–2022

Tape Subsystem User's Guide, Cray Research publication SG–2051

Tape Subsystem Administration, Cray Research publication SG–2307

NAME

nasa – Adds ASA carriage control characters for printing

SYNOPSIS

nasa [-t *tabspace*] [*file*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The nasa utility provides the function opposite to the asa(1) utility. It transforms typical UNICOS command text output to make it suitable for output to line printers that require ASA carriage control characters.

It processes either the *file* whose name is given as an argument or the standard input if you do not specify a file name. <tab> characters are expanded to the appropriate number of spaces to position a subsequent character at the next <tab> stop.

Other transformations performed are as follows:

line feed	Becomes <newline>, <space>
carriage return	Becomes <newline>, +
form feed	Becomes <newline>, 1

The nasa utility accepts the following option:

-t *tabspace* Specifies *tabspace* number of character <space>s between <tab> stops. Default is 8.

file File to be converted.

The nasa utility forces the first line to start on a new page by starting its output with a 1. <backspace> characters (ASCII code 8) are properly compensated for, to preserve the column position of subsequent <tab> characters, but the destination printer may not accept them.

NOTES

The nasa utility complements asa(1) only in that it converts raw data for printing on asa-style printers. nasa cannot undo the damage done to a file by inadvertently running it through asa(1).

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
system, secadm	Allowed to transform any input file. In a privileged administrator shell environment, shell-redirected I/O is not subject to file protections.

sysadm	Allowed to transform any input file subject to security label restrictions. Shell-redirected I/O is subject to security label restrictions.

If the PRIV_SU configuration option is enabled, the super user is allowed to transform any input file. Shell-redirected I/O on behalf of the super user is not subject to file protections.

EXAMPLES

Example 1: The following example is a standard shell script preparing the output of a job for printing on a line printer requiring carriage control:

```
(
    set -x                  # print command names as they are executed
    UNICOS commands
          .
          .
          .
    application | asa
    echo '\f'                  # form feed
          .
          .
          .
) 2>&1 | nasa > tracefile
```

Example 2: The following example prepares output to be printed and sends it to file datafile. Output is not sent to the printer.

```
ls -la | nasa >datafile
```

SEE ALSO

asa(1), expand(1), unexpand(1)

NAME

netstat – Displays network status

SYNOPSIS

```
/usr/ucb/netstat [-a] [-v] [-n] [-k] [-f address_family] [system] [core]
/usr/ucb/netstat [-A] [-v] [-n] [-k] [-f address_family] [system] [core]
/usr/ucb/netstat -g [-s] [-v] [-n] [-k] [-f address_family] [system] [core]
/usr/ucb/netstat -H [-f address_family] [system] [core]
/usr/ucb/netstat -i [-t] [-s] [-a] [-v] [-n] [-k] [-f address_family] [system] [core]
/usr/ucb/netstat -m [-s] [-v] [-n] [-k] [-f address_family] [system] [core]
/usr/ucb/netstat -q [-a] [-i] [-v] [-n] [-k] [-f address_family] [system] [core]
/usr/ucb/netstat -r [-v] [-n] [-k] [-A] [-f address_family] [system] [core]
/usr/ucb/netstat -s [-i] [-r] [-v] [-k] [-f address_family] [system] [core]
/usr/ucb/netstat [-I interface] [-n] [-k] [-f address_family] interval [system] [core]
```

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The netstat command symbolically displays the contents of various network-related data structures. The default display, which is for active sockets, shows the local and remote addresses, send and receive queue sizes (in bytes), protocol, and, optionally, the internal state of the protocol.

You can combine the following options and arguments with any of the main options (the list of main options follows this list):

-v
: Indicates verbose mode. You can repeat the -v option to obtain additional information. The option that accompanies the -v option determines the amount of additional information that can be displayed.

-n
: Shows Internet addresses as numbers. (Usually netstat interprets addresses and tries to display them symbolically.)

-k
: The -k option is provided for debugging purposes. It forces netstat to use /dev/kmem to obtain data, rather than use system calls.

-f *address_family*
: Limits statistics or address control block reports to those of the specified *address_family*. The address families that are recognized are inet (for AF_INET), trace (for AF_TRACE), and unix (for AF_UNIX).

system, *core*
: Obtains information from a system dump instead of the operating system. When a nonexistent system or core file is specified on the command line, netstat prints an error message and exits.

Main options:

`-a` Modifies the default display. The `-a` option indicates that all sockets will be shown (usually, sockets that server processes use are not shown).

If `-v` is used, the receive and send queue lengths are omitted, and the machine through which the connection is being routed is shown. If a second `-v` option is used, more information about the state of each TCP connection is printed (maximum segment size, send and receive windows, send and receive next sequence numbers, send and receive urgent pointers, unacknowledged send pointer, and retransmission sequence number).

`-A` Modifies the default display. With the `-A` option, the address of any associated protocol control blocks also is shown. This is used for debugging.

If `-v` is used, the receive and send queue lengths are omitted, and the machine through which the connection is being routed is shown. If you use a second `-v` option, more information about the state of each TCP connection is printed (maximum segment size, send and receive windows, send and receive next sequence numbers, send and receive urgent pointers, unacknowledged send pointer, the peers offered segment size, and retransmission sequence number).

`-g [-s]` Shows information related to multicast (group address) routing. By default, the `-g` option shows the Internet Protocol (IP) multicast virtual interface and routing tables. If the `-s` option is also present, it shows multicast routing statistics.

`-H` Shows statistics about sizes of packets that are read and written. These statistics are not subdivided for each interface. Packet size statistics are not maintained for CRAY J90 Ethernet, FDDI, or ATM interfaces; the `-H` option does not apply under these circumstances.

`-i [-t] [-s] [-a]`
Provides a table of cumulative statistics for transferred packets and errors for each interface that was autoconfigured. (Those interfaces that are statically configured into a system but are not located at boot time are not shown.) The network address (currently Internet-specific) of the interface and the maximum transmission unit (mtu) in bytes also are displayed. For secure systems, if the `-v` option is specified, the level and compartment range are displayed.

The `-t` option displays the timer value. If the interface is not using the timer field, it has a value of 0. With the `-v` option, the configuration and usage of the interface queues are displayed, and the flags are set for the interface. The queues are displayed in an *x/y* format; *x* is the number of packets currently on the queue, and *y* is the maximum number of packets that are allowed on the queue. The number of packets dropped because of the lack of queue space also is displayed.

The `-s` option displays the interface statistics in long format for each interface, and all of the interface statistics are printed with a short explanation.

The `-a` option shows multicast addresses currently in use. Multicast addresses are shown on separate lines following the interface address with which they are associated.

`-m [-s]` Shows statistics that the memory management routines record. (The network manages a private share of memory buffers called *mbufs* (pronounced em-bufs).) With one `-v` option, a list of the mbuf headers for all mbuf clusters in the system is shown. With two `-v` options, a list of the mbuf headers for all mbufs in the system, including those that are part of an mbuf cluster, is shown. With three `-v` options, a list of the mbuf headers for all mbuf clusters in the system is shown. The contents of the mbuf also are printed. Some types of mbufs are recognized and are printed as their structures; other types are printed as hexadecimal bytes. Besides the information shown by three `-v` options, four `-v` options show all mbufs in the system, including those that are part of an mbuf cluster. The `-s` option displays a histogram of mbuf request sizes.

`-q [-a] [-i]` Shows mbuf usage on the kernel IP input, IP reassembly, and raw input queues. If `-a` is specified, any mbufs on TCP reassembly queues also are shown.

If `-i` is specified, any mbufs on any of the interface output queues also are shown.

If `-v` is specified, a count and a list of the mbufs being used on each queue are printed. With `-vvk` options specified, up to the first 256 bytes of data are printed for each mbuf, relative to the offset in the mbuf. With `-vvvk` options, the first 256 bytes of data for each mbuf, or up to the offset, also are printed. It should be noted that the `-k` option is necessary to print the mbuf data.

Because of the transitory nature of these queues, if `-q` is used on a running system, it can easily become confused and give erroneous results. This option is best used on a core dump.

`-r [-k] [-A]` Indicates the available routes and status of each route. Each route consists of a destination host or network and a gateway to use for forwarding packets. The flags field indicates whether the route is for a host or a network, and whether an intermediate gateway or router is used (see flags below). If the route does not use a gateway, the gateway column may list the address for the interface used when sending to the destination, or may contain a link-layer address.

Network routes have an associated mask that specifies which parts of an address are matched. If destination is printed numerically and the mask is not the obvious value (e.g. if a route to an Internet Class B network has a mask other than 16 bits), the mask is indicated in one of two ways: If the mask is contiguous from the most-significant bit to the end, the usual case for subnets, a slash and the number of bits in the mask are appended to the network value. Otherwise, an ampersand (&) and a numeric representation of the mask are appended.

The flags field shows a collection of information about the route stored as binary choices. The individual flags are discussed in more detail in the `route`(8) and `route`(4P) man pages. The mapping between letters and flags is as follows:

1 Protocol specific routing flag #1
2 Protocol specific routing flag #2
B Discard all packets (during updates)
C Generate new routes on use
D Created dynamically (by redirect)
E Exclusive group ID list for route
G Destination requires forwarding by intermediary
H Host entry (default is net)
L Valid protocol to link address translation
M Modified dynamically (by redirect or mtu discovery)
m Route marked for no mtu discovery
N Route marked for no forwarding
R Host or net unreachable
S Manually added
T Route marked for exact or better type-of-service matching
U Route usable
X External daemon translates protocol to link address

Direct routes are created for each interface that is attached to the local host. The gateway field for direct routes shows the address of the outgoing interface. Some interfaces, such as Ethernet, also use link-level routes (e.g., see `arp`(8)). In those cases, the direct network route has the `C` flag set (for cloning), which causes individual host routes to be created on demand for hosts on that network. The host routes contain link-level *gateway* entries with their link-level addresses, and the `L` (link-level) flag is set. The `refcnt` field shows the current number of active uses of the route. Connection oriented protocols normally hold on to a single route for the duration of a connection while connectionless protocols obtain a route while sending to the same destination. The use field provides a count of the number of packets sent using that route. The interface entry indicates the network interface used for the route. If the `-v` option is also specified, additional information is provided. The *Admin MTU* field specifies the *mtu* the administrator set when the route was created using the `route`(8) command. The *Path MTU* field specifies the *mtu* discovery path *mtu* for the route. If *Path MTU* discovery has changed since the route was first installed, an asterisk (*) marks the path mtu. If the route was marked for *type-of-service*, security, or a group ID list, that information also is printed as appropriate. The group ID list is printed with a plus or minus (+/-) to indicate that the group ID list is inclusive (+) or exclusive (-). If the `-vv` option is specified, the routing metrics stored with the kernel are displayed. If the `-vvv` option is specified when examining a system dump, the kernel radix table is displayed. See the `net/radix.h` and `net/route.h` files for an explanation of the fields.

The `-A` option can be used when examining a system dump. The `-A` option causes the kernel memory addresses of route data structure to be printed. These addresses can be used to reconstruct the route radix tree.

`-s` [`-i`] [`-r`] Shows per-protocol statistics. To condense information on the screen, `netstat` does not display some entries if they have a 0 value. If `-r` also is used, statistics on the routing tables are given. If `-i` also is used, statistics on the interfaces are shown. If `-v` is used, all entries are displayed, whether or not they have a value of 0. With the `-iv` options, the interface statistics are gathered from the `ifcray` and `ifnet` data structures defined in the `crayif/ifc.h` and `net/if.h` files.

`-I` *interface* Shows information only about this interface; use with an *interval* argument as described with the `-n` option.

interval When `netstat` is invoked with an *interval* argument, it displays a running count of statistics that are related to network interfaces, and it pauses *interval* seconds before refreshing the screen. This display consists of a column that summarizes information for all interfaces, and a column that highlights the interface that has had the most traffic since the system was last rebooted. The first line of each screen of information summarizes information since the system was last rebooted, and subsequent lines of output show values that have accumulated during the preceding interval.

If a socket address specifies a network but no specific host address, address formats are of the form *host*`.`*port* or *network*`.`*port*. When known, the host and network addresses are displayed symbolically according to files `/etc/hosts` (see `hosts`(5)) and `/etc/networks` (see `networks`(5)), respectively. If a symbolic name for an address is unknown, or if the `-n` option is specified, the address is printed in the Internet dot format; see `inet`(3C) for more information about this format. Unspecified or *wildcard* addresses and ports appear as an asterisk (`*`).

BUGS

The errors are not well-defined. Collisions have no meaning for any of the interfaces that Cray Research supports. Some of the kernel tables are very transitory in nature, and `netstat` can become confused when used on a running system; either `???` is printed, or `netstat` might become hung up in an infinite loop.

EXAMPLES

Example 1: The following example shows the current routing entries for family 2 (`inet`).

```
$ netstat -r
Routing tables
Destination      Gateway            Flags     Refs     Use  Interface
default          core02-f20         UG         55   469940  fd0
loopback         localhost          UR          0        0  lo0
localhost        localhost          UH         10     5160  lo0
cray-hyp/24      cool               U           9    34665  np1
cray-fddi/24     link#18            UC          0        0  fd0
haze             0:0:77:85:29:df    UHL         0        6  fd0
gust-ip-fddi     0:0:a9:2:4:f       UHL         0        0  fd0
thunder          0:0:77:85:29:f7    UHL         2       61  fd0
cool-fddi        0:40:a6:d:4e:2     UHL         6     2510  lo0
sn5607-fddi      0:0:77:16:65:41    UHL         2   151945  fd0
ice-ip-fddi      0:0:a9:2:0:7e      UHL         0        0  fd0
ice              0:40:a6:d:4e:3     UHL         0        2  fd0
squall-ip-fddi   0:0:a9:2:1:e9      UHL         0        0  fd0
core01-f20       0:0:c:e:70:58      UHL         0        0  fd0
core02-f20       0:0:c:15:cc:ae     UHL         2        0  fd0
latte-alpha      core02-f20         UGHD        0      438  fd0
cool-030net      cool-030           U           7     2699  np0
lo0_multicast    localhost          UH          0        1  lo0
```

Example 2: The following example shows the currently configured interfaces.

```
$ netstat -ia
Name   Mtu   Network     Address      Ipkts   Ierrs Opkts   Oerrs
np0    16432 <Link>                    3780   0     3777    0
np0    16432 cool-030net cool-030      3780   0     3777    0
np1    16432 <Link>                   36097   0    35744    0
np1    16432 cray-hyp/24 cool         36097   0    35744    0
np2*   16432 <Link>                       0   0        0    0
np3*   16432 <Link>                       0   0        0    0
np4*   16432 <Link>                       0   0        0    0
np5*   16432 <Link>                       0   0        0    0
np6*   16432 <Link>                       0   0        0    0
np7*   16432 <Link>                       0   0        0    0
np8*   16432 <Link>                       0   0        0    0
np9*   16432 <Link>                       0   0        0    0
np10*  16432 <Link>                       0   0        0    0
np11*  16432 <Link>                       0   0        0    0
np12*  16432 <Link>                       0   0        0    0
np13*  16432 <Link>                       0   0        0    0
np14*  16432 <Link>                       0   0        0    0
np15*  16432 <Link>                       0   0        0    0
hi0*   16432 cray-hyp    sn131          7477   0     6163    0
fd0    4352  <Link>                  1039868   0   801090    1
fd0    4352  cray-fddi/2 cool-fddi   1039868   0   801090    1
lo0    65535 <Link>                    7670   0     7670    0
lo0    65535 loopback    localhost     7670   0     7670    0
```

FILES

`crayif/ifc.h`	Kernel include file that defines the `ifcray` data structure
`net/if.h`	Kernel include file that defines the `ifnet` data structure
`net/radix.h`	File that defines fields in the kernel radix table
`net/route.h`	File that defines fields in the kernel radix table

SEE ALSO

`inet`(3C) in the *UNICOS System Libraries Reference Manual*, Cray Research publication SR–2080

`hosts`(5), `networks`(5), `protocols`(5), `services`(5) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

UNICOS Networking Facilities Administrator's Guide, Cray Research publication SG–2304

NAME

newacct – Changes account ID

SYNOPSIS

```
newacct [-a[user]]
newacct [-l]
newacct [account-name]
```

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The newacct utility changes the account ID of the calling shell.

The newacct utility accepts the following options:

-a [*user*]
: Outputs the valid account names and account IDs for *user*. If *user* is not specified, the owner of the current process is the default *user*.

-l
: Prints the current account name.

account-name
: Validates whether you have permission to change the account ID to the requested value. If you omit this argument, newacct changes the account ID of the calling process to the default (primary) account ID of the user.

NOTES

If the fair-share scheduler has been activated on your system in the share by account ID mode, newacct also attaches the current session or job to the new account resource group. This may change the amount of system resources that you are allowed.

EXAMPLES

To list the account names and account IDs for jdoe, enter the following:

```
newacct -a jdoe
```

FILES

/etc/udb
: User validation file that contains user control limits

SEE ALSO

`acctid`(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

`getpwent`(3C), `id2nam`(3C), `putpwent`(3C) in the *UNICOS System Libraries Reference Manual*, Cray Research publication SR–2080

`passwd`(5), `udb`(5) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

NAME

newaliases – Rebuilds the sendmail(8) alias database

SYNOPSIS

```
newaliases
```

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The newaliases utility rebuilds the random access data base for the mail aliases file /usr/lib/aliases. It must be run each time this file is changed for the change to take effect.

The newaliases utility is identical to sendmail -bi.

The newaliases utility exits with a value of 0, if successful, and >0 when an error occurs.

NOTES

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
system, secadm	Shell-redirected I/O is not subject to security label restrictions.

If the PRIV_SU configuration option is enabled, for the super user, shell-redirected I/O is not subject to security label restrictions.

SEE ALSO

aliases(5) in the *UNICOS Administrator Commands Reference Manual*, Cray Research publication SR–2022

sendmail(8) in the *UNICOS Administrator Commands Reference Manual*, Cray Research publication SR–2022

UNICOS Networking Facilities Administrator's Guide, Cray Research publication SG–2304

NAME

newgrp – Changes to a new group

SYNOPSIS

newgrp [-l] [*group*]

Obsolescent version; may not be supported in future releases:

newgrp [-] [*group*]

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The newgrp utility changes a user's real and effective group ID. The user remains logged in and the current directory is unchanged. The user is always given a new shell, replacing the current shell with newgrp, regardless of whether it terminated successfully or due to an error condition (that is, unknown group).

Exported variables retain their values after invoking newgrp; however, all unexported variables are either reset to their default value or set to null. System environment variables (such as PS1, PS2, PATH, MAIL, and HOME), unless exported by the system or explicitly exported by the user, are reset to default values. For example, a user has a primary prompt string (PS1) other than $ (default) and has not exported PS1. After an invocation of newgrp, successful or not, the user's PS1 will be set to the default prompt string $. The shell command export (see sh(1)) is the method to export variables so that they retain their assigned value when invoking new shells.

With no arguments, newgrp changes the user's group IDs (real and effective) back to the group specified in the user's password file entry. This is a way to exit the effect of an earlier newgrp utility.

The newgrp utility accepts following option and operand:

-l
- (Obsolescent)
  Changes the environment to what would be expected if the user actually logged in again as a member of the new group.

group Identifies a group name from /etc/group or a numeric group ID. Both specify the group ID to which the real and effective group IDs are set.

If the group has a password and the user is not listed in `/etc/group` as being a member of that group, a password is required.

NOTES

On Cray Research systems, you cannot use the `newgrp` utility to create new files and directories under a different group. It also does not have an effect on UNICOS accounting or security features. The utility is provided for POSIX and XPG4 compliance.

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
`system`, `secadm`	Allowed to change to any group. In a privileged administrator shell environment, shell-redirected I/O is not subject to file protections.
`sysadm`	Allowed to change to any group. Shell-redirected I/O is subject to security label restrictions.

If the `PRIV_SU` configuration option is enabled, the super user is allowed to change to any group. Shell-redirected I/O on behalf of the super user is not subject to file protections.

EXIT STATUS

If `newgrp` succeeds in creating a new shell execution environment, whether or not the group identification was changed successfully, the exit status is the exit status of the shell. Otherwise, the `newgrp` utility exits with the following value:

>0 An error occurred.

FILES

`/etc/group`	System's group file
`/etc/passwd`	System's password file

SEE ALSO

`login`(1), `sh`(1),

`group`(5), `passwd`(5) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

NAME

news – Prints news items

SYNOPSIS

news [-a] [-n] [-s] [*items*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The news utility keeps users informed of current events. By convention, these events are described by files in the /usr/news directory.

When invoked without arguments, news prints the contents of all current files in /usr/news, most recent first, with each preceded by an appropriate header. news stores the ‘‘currency’’ time as the modification date of a file named .news_time in the user’s home directory (the identity of this directory is determined by environment variable $HOME); only files more recent than this currency time are considered ‘‘current.’’ When *items* is specified, the modification time of .news_time is not updated.

The news utility accepts the following options:

-a Prints all items, regardless of currency. In this case, the stored time is not changed.

-n Reports the names of the current items without printing their contents and without changing the stored time.

-s Reports the number of current items that exist, without printing their names or contents, and without changing the stored time.

items When specified, the modification time of .news_time is not updated. It is useful to include such an invocation of news in your .profile file, .cshrc file, or in the system’s /etc/profile or /etc/cshrc file to report whether there is news.

All other arguments are assumed to be specific news items to be printed. These items must match the names of files found in the /usr/news directory. If an item is not found, news continues with the next item in the list.

If an interrupt signal is received during the printing of a news item, printing stops and the next item is started. Another interrupt within 1 second of the first causes the program to terminate.

FILES

/etc/.cshrc	C shell default start-up file
/etc/profile	Korn shell default start-up file
/usr/news/*	Directory containing news item files

SEE ALSO

`cshrc`(5), `profile`(5) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

NAME

`nfsid` – Performs NFS authorization functions to NFS servers

SYNOPSIS

`nfsid` [[*options*] [*host* ...]] ...

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `nfsid` utility uses Kerberos authentication to present your credentials to an network file system (NFS) server, which allows your workstation to access files that reside in file systems exported only for Kerberos authenticated access. The default permissions for a client workstation that is attempting to access an NFS file system are those of the user "nobody," meaning that a program that is running on the workstation usually would not be allowed to modify any NFS server files. `nfsid` sends a Kerberos authentication ticket to the server, which records an association between a workstation's IP address and the user's user ID (UID).

The `nfsid` utility accepts the following arguments:

options Specifies the following arguments to modify the default behavior of the `nfsid` utility. All arguments are processed in the following order. Thus, if a `-u` option is followed by a `-m` option, the `-m` option takes precedence.

`-map` or `-m`	Sets the mapping function to `map user`. This is the default; it establishes a mapping to a server. It requires that the user be authenticated.
`-unmap` or `-u`	Sets the mapping function to `unmap user`. This argument removes a mapping from a server.
`-purge` or `-p`	Sets the mapping function to `purge host`. This argument removes all mappings associated with the user's host.
`-purgeuser` or `-r`	Sets the mapping function to `purge user`. This argument removes all mappings associated with the user on the host. It requires that the user be authenticated.
`-verbose` or `-v`	Displays verbose information about the mapping operation. This is the default.
`-quiet` or `-q`	Indicates that verbose information must not be displayed.
`-debug` or `-d`	Prints debugging information. This argument is not usually useful to users.

host Specifies a list of one or more hosts (either names or Internet addresses).

EXIT STATUS

If `nfsid` is executed with only one *host* argument, the exit status is one of the following:

Exit Status	Description
0	No error encountered
1	Bad arguments
3	Internal fatal error
10	Kerberos failure
11	Host communication failure
12	Authentication failure
13	No reserved ports available
21	Host name could not be resolved

If an error is encountered while one host in the list is being manipulated, processing continues with the other hosts. The exit status is returned after all hosts are attempted.

EXAMPLES

The following example establishes a mapping for the user on the host `CHARON.MIT.EDU` and deletes a mapping for the user on the host at Internet address `18.72.0.6`.

```
nfsid -m CHARON.MIT.EDU -u 18.72.0.6
```

LIMITATIONS

It is important to understand that the current implementation of NFS is not entirely secure when it uses the Kerberos authentication, including `nfsid`, `mountd`, and the UNICOS kernel. The current implementation assumes that the client hosts (those machines on which the users run `nfsid`) are single-user systems. Also, it assumes that no machine on the same subnet as the client host can be reconfigured to run with the same Internet address as the client host.

The kernel caches user ID/IP address pairs as directed by the `mountd` server, which responds to `nfsid` requests. When the client makes an NFS request to the server, the user ID/IP address pair from the NFS header is checked against the entries in the kernel. The problem with this method is that there is nothing to prevent any other user on the same client from creating the same NFS request, and filling in a false UID field. If this request is issued from the same IP address as that with which the client is registered in the server kernel, the server cannot detect that this request is not genuine. There is nothing in the NFS request itself that proves to the server that the request is genuine.

This is not to say that the `nfsid` scheme is worthless. It limits potential attacks to periods when the user ID/IP address pair is mapped in the server kernel. However, it is important to realize that the security that `nfsid` offers is limited when it is run from a multiuser host.

NAME

nice – Invokes a utility that has an altered scheduling priority

SYNOPSIS

nice [-n *increment*] *utility* [*arguments*]

Obsolescent version: may not be supported in future releases:

nice [-*increment*] *utility* [*arguments*]

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The nice utility invokes a *utility*, requesting that it be run with a different CPU scheduling priority.

If you omit *increment*, a default value of 10 is used.

The nice utility accepts the following options:

-n *increment*
-*increment* (Obsolescent)
: Specifies how the system scheduling priority of the executed utility is adjusted. The *increment* argument is a positive or negative decimal integer used to modify the system schedule priority of the executed utility.

  A positive *increment* value causes a lower or unchanged system scheduling priority. Any user can lower scheduling priority. A negative *increment* value causes higher or unchanged system scheduling priority. Only an appropriately authorized user can raise scheduling priority.

  If the requested *increment* would raise or lower the system scheduling priority of the executed utility beyond the nice value limits, then the limit whose value was exceeded is used. Nice values range from 0 (highest priority) through 39 (lowest priority).

utility [*arguments*]
: Specifies the utility or script to run at lowered priority.

NOTES

The csh(1) utility has a built-in nice utility that has slightly different characteristics. See csh(1).

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
`system`, `secadm`	Allowed to raise or lower the scheduling priority for any file.
`sysadm`	Allowed to raise or lower the scheduling priority for any file, subject to security label restrictions. Shell-redirected I/O is subject to security label restrictions.

If the `PRIV_SU` configuration option is enabled, the super user is allowed to raise or lower the scheduling priority for any file.

EXIT STATUS

If you invoke *utility*, the exit status of `nice` is the exit status of *utility*; otherwise, the exit status is one of the following:

1–125 An error occurred in the `nice` utility.

126 The utility specified by *utility* was found, but it cannot be invoked.

127 The utility specified by *utility* cannot be found.

SEE ALSO

`csh`(1), `renice`(1), `sh`(1)

`nice`(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

General UNICOS System Administration, Cray Research publication SG–2301

NAME

nl – Line-numbering filter

SYNOPSIS

nl [-b *type*] [-d *xx*] [-f *type*] [-h *type*] [-i *incr*] [-l *num*] [-n *format*] [-p] [-s *sep*] [-v *start*] [-w *width*] [*file*]

IMPLEMENTATION

All Cray Research systems

STANDARDS

XPG4

DESCRIPTION

The nl utility reads lines from *file* or standard input if *file* is not named and reproduces the lines on standard output. Lines are numbered on the left according to the command options in effect.

The nl utility views the text it reads in terms of logical pages. Line numbering is reset at the start of each logical page, which consists of a header, a body, and a footer section. Empty sections are valid. Different line-numbering options are independently available for header, body, and footer (for example, no numbering of header and footer lines while numbering blank lines only in the body).

The start of a logical page section is signaled by input lines containing nothing but the following delimiter character(s):

Line Contents	Start Of
\:\:\:	Header
\:\:	Body
\:	Footer

Unless otherwise specified, nl assumes that text being read is in a single logical page body.

Command options may appear in any order and may be intermingled with an optional file name. Only one file may be specified.

The nl utility accepts the following options:

-b *type* Specifies the logical page body lines to be numbered. Recognized types and their meanings are as follows:

- a Numbers all lines.
- t Numbers only lines with printable text; default.
- n Does not number lines.

p*string* Numbers only lines that contain the regular expression specified in *string*.

`-d` *xx* The delimiter characters specifying the start of a logical page section may be changed from the default characters (\:) to two user-specified characters. When only one character is entered, the second character remains the default character (:). To enter a backslash, use two backslashes.

`-f` *type* Same as `-b` *type* except for footer. Default for logical page footer is `n` (no lines numbered).

`-h` *type* Same as `-b` *type* except for header. Default *type* for logical page header is `n` (no lines numbered).

`-i` *incr* The increment value used to number logical page lines. Default is 1.

`-l` *num* The number of blank lines to be considered as one. For example, -l2 results in only the second adjacent blank being numbered (when the appropriate `-ha`, `-ba`, and/or `-fa` option is set). Default is 1.

`-n` *format* The line-numbering format. Default is `rn` (right-justified). Recognized *format* values are as follows:

`ln` Left-justified, leading zeros suppressed.

`rn` Right-justified, leading zeros suppressed.

`rz` Right-justified, leading zeros kept.

`-p` Does not restart numbering at logical page delimiters.

`-s` *sep* The character(s) used in separating the line number and the corresponding text line. Default *sep* is a tab.

`-v` *start* The initial value used to number logical page lines. Default is 1.

`-w` *width* The number of characters to be used for the line number. Default is 6.

file File to be filtered.

NOTES

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
`system`, `secadm`	Allowed to read from any input file. In a privileged administrator shell environment, shell-redirected I/O is not subject to file protections.
`sysadm`	Allowed to read from any input file subject to security label restrictions. Shell-redirected I/O is subject to security label restrictions.

If the `PRIV_SU` configuration option is enabled, the super user is allowed to read from any input file. Shell-redirected I/O on behalf of the super user is not subject to file protections.

EXIT STATUS

The `nl` utility exits with one of the following values:

0 Successful completion.

>0 An error occurred.

EXAMPLES

The following command line numbers `file1`, starting at line number 10, with an increment of 10:

```
nl -v10 -i10 -d!+ file1
```

The logical page delimiters are `!+`.

SEE ALSO

`pr`(1)

NAME

nlimit – Queries and modifies resource limits

SYNOPSIS

nlimit [-a *action*] [-c *cat*] [-i *id*] [-l *value*] [-q] *resource* [-t *type*]

nlimit [-q]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The nlimit utility allows the inquiry and setting of resource limits. Currently, only CPU time is supported as a valid resource.

The output contains all pertinent limit values, and when applicable, the action the system will take upon reaching a hard limit and the amount of resource consumed. In addition, when a limit value is unlimited, the output will print the message unlimited.

The nlimit utility accepts the following options:

-a *action* The action to take upon reaching a hard limit. When using the -a option, the -t option must be set to hard. Valid actions are as follows:

term or terminate	Terminate process
check or checkpoint	Checkpoint process and then terminate
null	Do not change action

-c *cat* Identifies which category of resource is to be queried or set. Acceptable values are as follows:

proc or process	Process limits
sess or session	Session limits
user or uid	User limits
sessprocs	Default process limits for the session

-i *id* The *pid*, *sid*, or *uid* correlating to the *cat* argument. Zero indicates the current *pid*, *sid*, or *uid*. The *cat* option of sessprocs requires an *sid*.

-l *value* The value of the new limit to be set. Acceptable values are as follows:

n	Positive integer. For CPU resources, this value represents seconds. (See the CAUTIONS section.)
unlimited	This specifies unlimited resources.
-1	Minus one. Do not change current limit. For use with changing the hard action but not changing the limit value.

-q Quiet mode. No headers will be printed with the output.

resource The resource you want to inquire about or change. Valid options are as follows:

`cpu` CPU resources. (See the CAUTIONS section.)

`-t` *type* Identifies the type of limit to be set or viewed. Valid options are as follows:

`abs` or `absolute`	Absolute limit (only the super user can set this limit)
`hard`	Hard limit
`soft`	Soft limit

Only an appropriately authorized user can increase resource limits. Only an appropriately authorized user can set the resource limits of another user, process, or session.

NOTES

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	**Action**
`system`, `secadm`	Allowed to raise or lower the resource limits of any user, process, or session.
`sysadm`	Allowed to raise or lower the resource limits of any user, process, or session, subject to security label restrictions. Shell-redirected I/O is subject to security label restrictions.

If the `PRIV_SU` configuration option is enabled, the super user is allowed to raise or lower the resource limits of any user, process, or session.

CAUTIONS

The CPU time limit does not apply when running as root.

EXAMPLES

The following examples show the inquiry and setting of resources. User input is shown in bold type.

Example 1: The following example shows the inquiry of all CPU limits.

```
$ nlimit cpu
RESOURCE   CATEGORY    ABSOLUTE        HARD        SOFT       ACTION    USED
-------------------------------------------------------------------------------
cpu:        process   unlimited         600         600   checkpoint       0
cpu:        session   unlimited       14400       14400    terminate     112
cpu:      sessprocs           0         600           0   checkpoint       0
cpu:            uid   unlimited   unlimited   unlimited    terminate   32935
```

Example 2: The following example shows the resetting of a CPU hard limit for process 1674.

```
$ nlimit -c process -i 1674 -t hard -l 2000 cpu
```

Example 3: The following example shows the inquiry of process 1674 (notice the change in hard limit value from the previous `nlimit` command).

```
$ nlimit -c process -i 1674 cpu
CATEGORY    ABSOLUTE        HARD        SOFT       ACTION    USED
-------------------------------------------------------------------
process    unlimited        2000         600   checkpoint      91
```

FUNCTIONAL DESCRIPTION

This section provides an overview of resource limits. Resource limit control is an extension of standard UNIX; therefore, it is not in effect unless explicitly specified. This ensures compatibility across UNIX systems and upward compatibility among UNICOS releases. To enable the resource limits feature, a user must explicitly use the `nlimit` utility, or the `nlimit`(3C) or `NLIMIT`(3F) library routines. CPU limits can be applied to sessions or processes running in foreground, background, or submitted through the Network Queuing Environment (NQE). Limits can also be applied against a user ID (UID).

RESOURCE LIMIT TYPES

There are three CPU limit types: absolute, hard, and soft. The system administrator defines the absolute limit; it cannot be modified by the user. Users define hard and soft limits. Each limit can be applied to three different categories: user, session, and process. Another category, `sessprocs`, can only have a hard limit defined.

Absolute Limit Type

The absolute limit type is the system administrator-defined limit and is the maximum amount of a system resource available to the user. This limit could be the maximum of the system resource or some system administrator-defined value less than the maximum system resource. Reaching this limit will cause the user program to terminate.

Hard Limit Type

The hard limit is user-defined, and can be set equal to or less than the absolute limit value. Associated with the hard limit is a user-definable default action for when the hard limit is exceeded. The action is either "terminate" or "checkpoint and terminate." By default, if a user process reaches a hard limit, the program will terminate. The hard limit action determines whether the program will attempt to be checkpointed before termination.

Soft Limit Type

The soft limit is user-defined and can be set equal to or less than the hard limit value. Upon hitting a soft limit, a `SIGINFO` signal is sent to the program. The user can register the program to catch the signal, using `getinfo`(2), query the reason for the signal, and then take some action based on the reason for the signal.

RESOURCE LIMIT ACTIONS

This section describes the actions taken when the absolute, hard, and soft limits are reached for processes, sessions, and users.

Process Absolute Limit

If the absolute limit is exceeded on a process, that process is sent the `SIGCPULIM` signal followed by the `SIGKILL` signal and is terminated.

Session Absolute Limit

If the absolute limit is exceeded on a session, then all processes related to that session are sent the `SIGCPULIM` signal followed by the `SIGKILL` signal and are terminated. If the user is logged on interactively and their login shell is part of the session that exceeded the limit, the login shell will be terminated and the user will be logged off the system.

User Absolute Limit

If the absolute limit is exceeded on a user, then all processes owned by that user are sent the `SIGCPULIM` signal followed by the `SIGKILL` signal and are terminated. If the user is logged on interactively, the user will be logged off the system. User access to the system will be denied if the user absolute limit has been reached. The system administrator must either raise the absolute limit or set the user's accumulated CPU to a lower value.

Process Hard Limit

If the hard limit is exceeded on a process, the default hard action is checked. If the default hard action is to terminate, the process is sent the `SIGCPULIM` signal followed by the `SIGKILL` signal and is terminated. If the default hard action is to checkpoint and terminate, then the system attempts to checkpoint that process before termination. If a checkpoint file cannot be created, a standard core file will be created. For Fortran users wanting a checkpoint file by using the hard action of checkpoint, set the environment variable `TRACEBK` to 0 prior to running your application.

Session Hard Limit

If the hard limit is exceeded on a session, the default hard action is checked. If the default hard action is to terminate, all processes related to that session are sent the `SIGCPULIM` signal followed by the `SIGKILL` and are terminated. If the default hard action is to checkpoint and terminate, then the system attempts to checkpoint all processes related to the session before termination. Because checkpoint creates a file called `core`, it is possible when checkpointing multiple processes in the same directory to overwrite previous `core` files. If the user is logged on interactively and their login shell is part of the session that exceeded the limit, the login shell will be terminated and the user will be logged off the system.

`sessprocs` Hard Limit

The `sessprocs` limit lets the user set up default hard process limits per session. If a `sessprocs` limit is set, all processes created in this session have the hard and soft limit set to the `sessprocs` limit. Two values can be set by `sessprocs`: the hard limit and the hard limit action. Because this sets up process hard limits, the action resulting from hitting a `sessprocs` limit is the same as hitting a process hard limit.

User Hard Limit

If the hard limit is exceeded on the user category, the default hard action is checked. If the default hard action is to terminate, all processes owned by that user are sent the SIGCPULIM signal followed by the SIGKILL signal and are terminated. If the default hard action is to checkpoint and terminate, the system attempts to checkpoint all processes owned by the user before termination. If the user is logged on interactively, the user will then be logged off the system. User hard limits are valid only as long as the user has an active process on the system.

Process Soft Limit

If the soft limit is exceeded on a process, the SIGINFO signal is sent to that process. This SIGINFO signal can be trapped by the user. It is up to the user to catch the SIGINFO signal by registering for the receipt of this signal. If the user has registered for the SIGINFO signal, control is returned to the user program. If the user has not registered for the SIGINFO signal, no signal is sent and the process continues to execute until the hard limit is reached or the session completes. There could be multiple reasons for the posting of a SIGINFO signal. Therefore, after the soft limit signal has been received, the user should query the reason for the soft signal by using the getinfo(2) system call.

Session Soft Limit

If the soft limit is exceeded on a session, all processes related to that session that have registered for the SIGINFO signal are sent the SIGINFO signal.

User Soft Limit

If the soft limit is exceeded on the user category, all processes owned by that user having registered for SIGINFO are sent the signal. If the user is logged on interactively, the user will not notice this limit being reached. User soft limits are valid only as long as the user has an active process on the system.

DEFAULT LIMITS

Absolute limits (process, session, and UID) are set at login time from values in the user database (UDB). The UID limits are in effect only if the fair-share scheduling feature is enabled. The process, session, and UID hard and soft limits are set equal to the absolute limit at login time. The sessprocs hard limit value is set equal to the process absolute limit. The default hard action is to terminate.

TYPE CHECKING PRECEDENCE

The absolute limit takes precedence over the hard and soft limit. The hard limit takes precedence over the soft limit if they are set equal. This means if all limits are the same (soft = hard = absolute) and the limits are exceeded, the absolute action will be followed (terminate). If a hard limit is specified but no soft limit, the soft limit is set equal to the hard limit. If no default hard action is specified, the default hard action will be to terminate.

CATEGORY CHECKING PRECEDENCE

Process limits will be checked before session limits and session limits checked before user limits.

EXPLICITLY SETTING LIMITS

Two ways exist to view, set, and modify limits: through the `nlimit`(1) utility and through the `nlimit`(3C) and `NLIMIT`(3F) library interface. The hard and soft limit values can be raised and lowered as long as the limit does not exceed the absolute limit.

IMPLICITLY SETTING DEFAULT LIMITS UPON LOGIN

Although the user must explicitly define the hard and soft limits for them to be in effect, the `nlimit` utility can be added to the `.login`, `.cshrc`, or `.profile` scripts. This lets the user set up default limits without having to explicitly define them upon every login. Another means for implicitly setting limits is to use the `sessprocs` category. The setting of a `sessprocs` hard limit implies that all processes created in the session will have this limit.

LIMIT CASCADING

Any new process created will inherit their parents session and UID limits. Process limits will be inherited either from their parent or from the `sessprocs` value. If the `sessprocs` limit is unlimited, the child will inherit the parent process limits. If the `sessprocs` limit is not unlimited, the child will inherit the `sessprocs` limit.

RESTARTING A LIMIT-INDUCED RESTART FILE

To restart a checkpoint file that was created due to reaching a hard limit, the current process limits must be greater than the process limits of the checkpoint file.

ACCUMULATED RESOURCE USAGE

The `nlimit` utility and the `nlimit`(3C) and `NLIMIT`(3F) library routines can return the current amount of CPU time accumulated for a process, session, and user.

SEE ALSO

`limit`(1)

`getlim`(2), `setlim`(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

`NLIMIT`(3F) in the *Application Programmer's Library Reference Manual*, Cray Research publication SR–2165

`nlimit`(3C) in the *UNICOS System Libraries Reference Manual*, Cray Research publication SR–2080

UNICOS Shells Ready Reference, Cray Research publication SQ–2116

NAME

nm – Prints name list

SYNOPSIS

nm [-a] [-A] [-c] [-e] [-f] [-g] [-L] [-m] [-n] [-o] [-P] [-t *format*] [-u] [-v] [-V] [-x] *files*

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4
AT&T extensions (-e and -V options)
BSD extensions (-n)
CRI extensions (-a, -c, -f, and -m options)

DESCRIPTION

The nm utility prints the name list (entries and external files) of each object *file* specified on the command line.

There are two forms of output. The -P flag results in the POSIX standard portable output, described later. Absent that flag, the traditional UNICOS output is produced. Each symbol name is preceded by its value or its size (blanks if undefined) and by its type (see the end of this section).

The nm utility accepts the following options:

-a Prints the variable names and the label names found in module or common block debug tables.

-A Causes the full pathname or library name to be prepended to the symbol name.

-c Adds secondary identification characters to reflect Cray Research enhancements to relocatable table formats. For a description of these characters, see the end of this section.

-e Prints only the defined symbols.

-f Produces full output. Writes redundant synbols that are normally suppressed.

-g Prints only global symbols. If the -g option is not present, the local block names (for example, X$DATA) are printed.

-L **What is this?**

-m Appends the module name to the file name. (Not effective if the -P option is selected.)

-n Sorts numerically rather than alphabetically.

-o Writes numeric values in octal (equivalent to -t o).

-P Prints in the POSIX standard portable format.

`-t` *format*
Writes each numeric value in the specified format. The format is dependent on the single character used as the *format* option-argument:

`d` Writes the offset in decimal.

`o` Writes the offset in octal.

`x` Writes the offset in hexadecimal.

`-u` Prints only undefined symbols.

`-v` Sorts output by value instead of alphabetically.

`-V` Prints the version of the program on `stderr`.

`-x` Prints numeric values in hexidecimal (equivalent to `-t x`).

In the POSIX portable output, the symbol name is followed by a character describing its type (see below), its value and, if appropriate, by its size. Both the number and size are represented in the number base determined by the `-t` flag: `-t d` produces decimal, `-t o` octal, and `-t x` hexadecimal. If `-t` is not specified, the number and size are represented in hexadecimal.

If the `-A` flag is specified, the symbol name is preceded by the file name, followed by a library member name enclosed in square brackets if appropriate, followed by a colon.

Symbols are written sorted by symbol name, collated as determined by the current locale, unless the `-v` flag is specified, in which case symbols are sorted by value.

If the `-u` flag is specified, only undefined symbols are written.

Symbol types are:

`A` Global absolute
`B` Global bss segment symbol
`b` Local bss segment symbol
`C` Common block
`D` Global data segment symbol
`d` Local data segment symbol
`L` Block resides in local memory
`T` Global text segment symbol
`t` Local text segment symbol
`U` Undefined

If the `-c` option was specified, the following characters describe additional information on relocatable formats:

`b` A bss block or common block.
`e` A common block that is also an entry point.
`s` A soft external.
`x` An external passed as an address by a Fortran routine.
`z` A bss block or common block that may be preset to 0.

EXIT STATUS

The nm utility exits with one of the following values:

0 Successful completion.

>0 An error occurred.

EXAMPLES

Example 1: The following example searches two libraries for a specific global symbol:

```
nm -go /lib/lib1.a /lib/lib2.a | grep symbol_name
```

Example 2: This example lists all the references that must be satisfied from libraries after the files in a development directory have been successfully compiled into relocatables:

```
nm -g *.o | sort | uniq
```

SEE ALSO

ar(1) for archive and library maintainer for portable archives
bld(1) to maintain relocatable libraries
lorder(1) to find ordering relation for an object library

ar(5) for archive file format
relo(5) for relocatable object table format under UNICOS in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014.

NAME

nmake – Maintains and updates programs

SYNOPSIS

nmake [-b] [-c] [-d *n*] [-f *makefile*] [-g *makefile*] [-h] [-i] [-j *n*] [-k] [-l] [-m] [-n] [-o] [-q] [-r] [-s] [-t] [-u] [-v] [-x] [-A] [-D*name*[=*value*]] [-F] [I*directory*] [-M] [-O] [-R] [-S] [-T] [-U*name*] [-V] [-X] [*targets*]

nmake -b -f -l -r *base*.ms

nmake -l -f /dev/null

IMPLEMENTATION

Cray PVP systems

DESCRIPTION

This man page describes the function of /usr/bin/nmake. The nmake utility reads the input *makefile* and executes shell commands (see sh(1)) to update one or more *targets*. Each makefile contains a sequence of entries that specify dependencies. An entry consists of a nonempty list of targets and a list of dependency rules for the target, separated by a colon (for example, *target : dependency rules*). A target is updated if at least one of its dependencies was modified since the target was last modified or if the target does not exist.

The nmake utility interacts only with sh(1); it does not function with csh(1).

The makefiles are passed through cpp(1) or /usr/lib/make/cpp (for system builds) and are compiled into object files before any actions are taken. If a makefile was not modified since the last nmake, the corresponding object file is used. Each nmake object file is placed in *base*.mo; *base* is the base name (suffix deleted) of the corresponding makefile.

You can find further guidelines for using this utility for system builds in *UNICOS Configuration Administrator's Guide*, Cray Research publication SG–2303.

The nmake utility accepts the following options.

Built-in attributes that match the regular expression:

```
.[.A-Z][.A-Z0-9]*
```

User attributes that match the regular expression:

```
.[.A-Z][.a-z0-9]*
```

Intermediate targets that match the regular expression:

```
.[.a-z][.a-z0-9]*
```

Use `${!:-$$}` to generate temporary file names.

Options are preceded by a `-` or `+` and can appear anywhere on the command line. `+` turns off the corresponding option. Command-line options override options specified in the makefile. Options not in the following list can be interpreted by local versions of the built-in rules. The options are as follows:

`-b` Does not use the built-in rules.

`-c` Forces the makefiles to be compiled into one `nmake` object file. `nmake` exits after the files are compiled.

`-d` *n* Provides a step-by-step dump of `nmake`'s actions. The optional number argument, *n*, selects the dump level; higher levels produce more output. The debug levels and the objects that are traced are as follows:

0 No trace
1 Basic `nmake` trace
2 Major actions
3 Coshell messages
4 State variables, `M` information
5 Makefile input lines
6 Directory and archive scan information
7 Detailed hash table information
8 Built-in rule input lines
9 Lexical analyzer states

`-f` *makefile* Reads the descriptions in *makefile*. If the `-f` option is not specified, the files specified by the dependencies of the `.MAKEFILES` rule are tried in order from left to right. If *makefile* is `-`, the standard input is read. More than one `-f` option can appear; the files are read in order from left to right. If no `-f` option is specified and no default *makefile* is found, suffix `.mk` is appended to the base name of the first argument in the *makefile* dependency files and this file is read. If no *makefile* can be found, an error occurs.

`-g` *makefile* Treats *makefile* like a global makefile, similar to the `-f` option. This means that the default *makefile*s are still tried if no `-f` option appears.

`-h` Does not automatically search files with suffixes marked by `.SEARCH` for `#include` header file dependencies.

`-i` Ignores command error codes from the shell.

`-j` *n* Specifies that `nmake` can execute up to *n* command update jobs concurrently. If omitted, *n* defaults to 1 (when using the install tool, the default is 3).

`-k` If the update commands for the current target return nonzero status, `nmake` continues working on targets that do not depend on the current target. This is the same functionality as the `.DONTCARE` built-in rule.

`-l` Lists variable and rule definitions after the *makefile*s are read. The targets are neither checked nor updated, only listed.

`-m` Moves each target to the directory of the corresponding primary source file, which is determined by using the implicit suffix rules. Usually, targets are placed in the current directory. Archive members and targets with explicit directory names are not moved.

`-n` Traces and prints, but does not execute, the target update commands. This option also inhibits *makefile* compilations.

`-o` Outputs intermediate dependency information.

`-q` Does not update any targets but exits with a value of 0 if the targets are up-to-date; otherwise, it exits with a value of –1.

`-r` Lists the detailed status of each rule after all targets are made. If the `-l` option is also set, the listing occurs before any targets are made and `nmake` exits without making any targets.

`-s` Executes but does not print the update commands. This is the same functionality as the `.SILENT` built-in rule.

`-t` Touches the modify date of targets, bypassing the update commands. Only existing targets are touched. Targets that have state variable dependencies can be touched only by use of this option. This is the same functionality as the `.TOUCH` built-in rule.

`-u` Does not force file bindings to be unique and does not warn about duplicate files.

`-v` Lists variable assignments. Useful in conjunction with the `-d` option.

`-x` Does not check implicit file dependencies generated by a previous `nmake`.

`-A` Accepts any existing targets as being up-to-date, even when the targets have state variable dependencies; however, targets out-of-date with file dependencies are always updated.

`-D`*name*[=*value*]
Defines *name* as though defined by a `#define` directive. If [=*value*] is omitted, *name* is defined as 1. No space is allowed between `-D` and *name*.

`-F` Forces all targets to be updated.

`-I`*directory* Passes parameters to `cpp` when *makefile*s are read. `nmake` first searches for `#include` files that are enclosed in quotation marks and do not begin with a / symbol in the current directory; the directories specified by the `-I` option are searched next, and then the directories specified on the standard list are searched. No space is allowed between `-I` and *directory*.

`-M` Causes `nmake` to act as if no command-line target was made.

`-O` Overrides explicit command update blocks by applying only implicit command update blocks.

`-R` Forces the input makefiles to be read rather than loading the corresponding `nmake` object files.

`-S` Ignores previous state variable definitions.

`-T` Enables testing code. Do not use this option unless you are familiar with the source code and you know what the current test is.

`-U`*name* Removes any initial definition of *name*; *name* is a predefined reserved name. The `-U` option overrides the `-D` option. The `#assert` directive makes reserved preprocessor names obsolete. No space is allowed between `-U` and *name*.

`-V` Lists the current `nmake` version.

`-X` Clears the special rules `.SOURCE.a` and `.SOURCE.h`.

targets The file to be written to.

`-b -f -l -r`
Lists the contents of the state variable file.

`-l -f /dev/null`
Lists the default rule and variable definitions.

Makefiles

You can specify commands in a makefile by starting the command line with <TAB> or <SPACE>, or by enclosing the command with braces { }; neither the space nor the tab can precede an initial target name.

Both the list of dependency rules and the shell command list can be empty. Comments are the same as in the C language. A # in column 1 is interpreted by `cpp`; otherwise, `nmake` treats text (including a space or a tab) between the # symbol and the newline character as a comment.

A dependency rule can appear as a target in more than one entry; the dependencies for successive entries are simply appended. Only one shell command list may be specified for a given target. Rule names that contain the special characters `:`, `{`, `}`, #, `=`, and `+` must be enclosed in double quotation marks. The dependency rules for a target are scanned in order from left to right.

Rule Binding

The `nmake` utility binds each rule to either a file or a state variable in the process of updating targets. Rule names are bound to file names by the dependencies of the built-in `.SOURCE` and `.SOURCE.`*x* rules (see the Special Rules subsection). The dependencies of these rules are the directories that are to be scanned when searching for files. `nmake` warns when the same file is found in more than one directory, but it continues with the first file found.

A *state variable* is a variable whose time of modification and whose definition was stored from one invocation of `nmake` to the next. State variables have two basic forms: `(`*variable*`)` is a makefile variable (see the Variables subsection) and *file*`(`*variable*`)` is a variable that corresponds to a `#define` statement in *file*. The following command line specifies that `x.o` (the target) depends on the definition of the `DEBUG` variable from the `header.h` file and that it also depends on the definition of `MACHINE` (a variable) from the current makefile:

```
x.o : header.h(DEBUG) (MACHINE)
```

Usually, state variables are not included in automatic variable expansions (`nmake` maintains automatic variables; see the Automatic Variables subsection). State variable definitions are stored in *base*`.ms` in the current directory; *base* is the name of the first makefile in the argument list.

If a state variable is defined by a `#define` statement that is part of a conditional `#if, #ifdef`, or `#ifndef`, only the first encountered definition is used.

Variables

Entries in makefiles can contain variable definitions of the following form:

variable = value

In this example, *value* can be composed of several elements. Subsequent appearances of `$(`*variable*`)` are replaced by *value*. `$$(`*variable*`)` is replaced by `$(`*variable*); otherwise, `$` is passed untouched. The *value* is not expanded until `$(`*variable*`)` is encountered. If *value* contains another variable, the contained variable is expanded before *value* is determined.

Use `:=` instead of `=` to expand the *value* immediately; use `+=` to expand and append *value* to the current value of *variable*. You can modify command-line variable assignments only by using the `+=` assignment operator.

Variable definitions come from many sources. The precedence order (highest to lowest) is as follows:

1. Automatic definitions
2. Dynamic definitions
3. Command-line definitions
4. Makefile definitions
5. Environment definitions
6. Built-in definitions

Automatic definitions are maintained by `nmake` (see the Automatic Variables subsection). *Dynamic definitions* occur when targets are updated. *Command-line* arguments of the form *variable=value* are considered command-line definitions. *Makefile definitions* are included in the individual makefiles that are used with the `nmake` utility. *Environment definitions* are read from the inherited environment (see `env`(1)). *Built-in definitions* associate attributes with targets and dependency rules.

Variables in target and dependency names are expanded once when the makefile is read and once when the target or dependency rule is bound. Variables in shell command lists are expanded each time the commands are executed. If `$` is the first character on a line, the corresponding variable is always expanded when the makefile is read. This allows conditional makefile input, using edit operators such as `T=N` or `T=V`.

A variable name is a sequence of letters, digits, underscores, and periods. Variables with names that contain a period (`.`) should not conflict with environment variables set by `sh`(1); typically, such variables are used in built-in rule specifications. To avoid conflicts with the built-in rules, do not define uppercase variable names with a `.` as the first character.

Editing Variable Values

You can edit variable values during expansion by using the built-in edit operators. The general syntax for the editing operator is `$(`*variable*`:`*op*`[=`*arg*`]:...)`. The operators are applied to each space-separated element of the expanded *value* in which the newline character is treated as a separate element.

You can edit the following components of variable file names:

`M` *machine*	Edits all characters up to and including the last `!` (inserts *null* if no `!` appears).
`D` *directory*	Edits all characters after the last `!`, up to and including the last `/` (inserts *null* if no `/` appears).
`P` *prefix*	Edits all characters after the last `/`, up to and including the first `.` symbol. (inserts *null* if less than two `.` appear or if `.` is the first character).
`B` *base*	Edits all characters after the first `.` symbol, up to but not including the last `.` symbol.
`S` *suffix*	Edits all characters from the last `.` symbol to the end (inserts *null* if no `.` appears).

You can edit variables by using the `$(`*variable*`:`*edit*`)` syntax, in which *edit* is a concatenation of *c:* or *c*=*new* causes *variable* to be edited. *c* specifies the `M`, `D`, `P`, `B`, or `S` component, and *new* specifies a new value for the component.

If component letters are omitted, the corresponding component is deleted from the substituted value. The following variable name is used for substitution in the following examples:

```
FILES = a.c dir/s.x.c bozo!.profile
```

Following are examples of the use of the `B` and the `M` edit operators:

```
$(FILES:B:S=.o)      Results in a.o x.o .profile.o
$(FILES:M)           Results in bozo!
```

Other edit operators and examples based on the previous variable name are as follows:

`A[!]=a[|`*b...*`]`

Selects components that [do not] have the user attribute *a* [or *b...*]. An example of this is as follows:

```
$(FILES:A=.c:)       Results in a.c dir/s.x.c
$(FILES:A!=.c:)      Results in a.c dir/s.x.c boxo!.profile
```

`C`<*delim*>*old*<*delim*>*new*<*delim*>

Substitutes every occurrence of string *old* with string *new* for each delimiter-separated element. <*delim*> can be any delimiter character. If `^` is the first character in *old*, it represents the beginning of each element of *variable*'s expanded value; if # is the last character, it represents the end of each element. If *old* is null, `&` in *new* is replaced by the current element. `C/` may be abbreviated as `/`. An example of this is the following substitution:

```
$(FILES:c/ /,/:)      Results in a.c,dir/s.x.c,bozo!.profile
```

`E=`*message* Outputs *message* as an error on `stderr`, and `nmake` exits with nonzero exit status. Messages are in the following format:

`nmake:`*(error)message*

`F=`*format* Expands tokens according to *format*, which can be a concatenation of the following:

`L` Converts the token to lowercase.

`U` Converts the token to uppercase, as in the following example:

`$(FILES:F=U%.3s:)` Results in `A.C DIR BOZ`

This example uses the `%[`−`][`*n*`][.`*m*`]`*c* `printf`(3C) style formatting; *c* can be any of the following print conventions: `s`, `d`, `o`, `x`, and `u`.

`G=.`*s* Selects elements that can generate files with the *.s* suffix by using the implicit rules (see the Implicit Rules subsection for details); that is, if the implicit rule **.s : *.y* exists, token *x.y* is selected.

`I=`*message* Displays *message* on `stderr`; these are informational messages that do not interrupt `nmake` processing. Messages are in the following format:

`nmake:`*(information)message*

`N[ ! ]=`*pattern* Selects only tokens [not] matching the shell file match *pattern*. An example of this is the following substitution:

`$(FILES:N=*.c)` Results in `a.c dir/s.x.c`

`T=`*type*
`T=`*type*`[ ! ]=`*pattern*
`T=`*type*`?`*return*

Selects tokens specified by *type*. `[ ! ]=`*pattern* specifies that the substituted tokens must also [not] match the *pattern*; if there is no match, a null string is returned; otherwise, *type* can be one of the following:

`A` Selects each token that can be bound to an archive.

`B` Translates multiple space-character sequences to a single space.

`D` Expands each token that can be bound to a *state variable*, using the state variable definition. The expanded definitions can be used as arguments to `cc`(1).

`E` Similar to `T=D` except that the expanded definitions are of the format *x=y*; *x* is the state variable, and *y* is its value.

`F` Selects each token that can be bound to a file by using the *bound* file name.

N Expands the null string if *variable* has a null value; otherwise, # is expanded. This can be used to specify conditional makefile input.

O Selects each token that is bound neither to a file nor to a *state variable*.

S Selects each token that can be bound to a *state variable*.

T Selects each token that appears as a target in the *makefiles* input.

V Expands the null string if *variable* has a nonnull value; otherwise, # is expanded. This can be used to specify conditional makefile input. The *value* of *variable* is substituted without expansion.

W=*message* Outputs *message* as a warning on `stderr`. Messages are of the format:

`nmake:`*(warning)message*

X=*cross* Expands the cross product of the tokens in *variable* and the tokens in *cross*.

The following examples show the use of the `T=`*type* edit operator. The original variables, before substitution, are as follows:

```
FILES=x.c (DEBUG) (TEST) libc.a
TEST =
DEBUG = 1
```

The variables after substitution are as follows:

```
$(FILES:T=D:)      Results in -DDEBUG
$(FILES:T=F:)      Results in x.c  /lib/libc.a
$(FILES:T=S:)      Results in (DEBUG) (TEST)
$(TEST:T=V:)       Results in #
$(FILES:T=V:)      Results in null string
$(FILES:T=E:)      Results in DEBUG=1 TEST=
```

Only one of each operator can occur for each edit expansion. The operators are evaluated in the following order: `X`, `V`, `G`, `N`, `A`, `T`, `C`, `M`, `D`, `P`, `B`, `S`, and `F`.

Using the $(*var1* | *var2*...) syntax causes the value of the first nonnull variable (left to right) to be substituted. If the last variable name is enclosed in double quotation marks (`""`), and if all preceding variables have null values, this string is used as the value. The standard `\` character constants are interpreted within the string. $(*file(variable)*) causes the value of state variable *file(variable)* to be substituted as in the following example:

```
FILES = a.h b.h c.h x.c y.c z.c
HEADERS = $(FILES:N=*.h)

$(HEADERS:/^/-I/:) becomes -Ia.h -Ib.h -Ic.h
$(HEADERS:/ /:/:) becomes a.h:b.h:c.h
```

Automatic Variables

The nmake utility maintains the following variables:

`$(-)` The concatenation of the option flags presented to nmake (with the preceding –).

`$(-`*x*`)` This variable expands to 1 if the option is set; otherwise, it is null.

`$(=)` The list of variable assignments in the nmake utility arguments.

`$(?)` The list of nmake utility arguments that were not made yet. After this expansion, all arguments are treated as if they were already made. `$(?)` and `$(=)` are useful in recursive nmake calls.

`$(<)` The current target name.

`$(>)` The list of all file dependencies of the current target that are out-of-date with the target.

`$(*)` The list of all file dependencies of the current target.

`$(&)` The list of all file and state variable dependencies of the current target.

`$(%)` If the current target is a state variable, `$(%)` is the state variable value; otherwise, it is the unbound rule name.

`$(@)` The precommand list for the current target.

`$(#)` The postcommand list for the current target.

`$(MAKE)` The name of the current nmake program.

`$(MAKEFILE)`
The name of the first nonglobal makefile.

Each successive occurrence of <, >, *, &, %, @, or # causes the parent of the current target to be accessed. For example, `$(<<)` is the name of the parent of the current target, and `$(**)` is the list of all of its dependencies. `$(c`*rule*`)` (with c referring to one of the automatic variables) accesses information for *rule* rather than the current target. Also, `$(+`*c*`)` accesses information for the first dependency of the current target. The state variables and the .NOTOUCH, .POST, and .USE rules are not included in <, >, or * automatic variable substitutions.

Following is an example of this use of automatic variables:

```
PROG::g.y  g.l  g.c
   echo (rule $(<*.o)
```

This results in:

```
g.o  g.y  g.l
```

Operators

The built-in rules for nmake also contain operator definitions. An operator specifies actions to be taken when makefiles are read and often allow abbreviated makefile specifications. The :: operator is similar to the : edit operator in specifying dependencies, except that :: specifies source dependencies.

Any ld(1) library flags and libraries with suffix .a can be placed in :: dependency lists only if the directory containing the .a libraries is defined by the .SOURCE.a rule. By default, .SOURCE.a is set to /lib /usr/lib. The LDFLAGS variable (-L) is ignored because nmake expands -lx to libx.a.

Option Generation

The built-in rules automatically generate the proper -D and -I options of cc(1) in the $(CCFLAGS) variable. The -D options are generated from state variable dependencies, and the -I options are generated from the dependencies of the .SOURCE.h rule. State variable dependencies that are specified using the :: operator apply to all dependencies of the corresponding target (global dependencies); otherwise, the dependencies apply only to the individual target of the operator.

The following makefile specifies that program depends on two files, a.o and b.o, and that they in turn depend on .c files and the common file header.h:

```
program : a.o b.o
   cc a.o b.o lib.a -lm -o program
a.o : header.h a.c
   cc -c a.c
b.o : header.h b.c
   cc -c b.c
```

Implicit Rules

The nmake utility infers dependencies for files that do not have explicit update commands. For example, a .c file can be inferred as a dependency for a .o file and can be compiled to produce the .o file.

The rule for creating a file with suffix .s2 that depends on a file (with the same *base* name) with suffix .s1 is specified as an entry for the target rule:

```
*.s2 : .s1
```

The suffixes .s2 and .s1 are appended automatically to .SUFFIXES if they were not specified previously. Any dependencies following *.s1 are transferred to each implicit target when the implicit rule is applied. For example, a rule for making optimized .o files from *.c files is the following:

```
*.o : *.c (CC) (CCFLAGS)
$(CC) $(CCFLAGS) -c -o $(<) $(>)
```

If the current target is a.o, nmake infers the following rule from the *.o : *.c rule:

```
a.o:a.c
cc -c -O -o a.o a.c
```

Using the default *.o : *.c rule, the example can be stated more briefly:

```
program : a.o b.o
  $(CC) $(*) lib.a -lm -o $(<)
a.o b.o : header.h
```

The rule for creating a file with no suffix from a similarly named file with suffix `.s` is specified as an entry for the target rule `* : *.s`. The rule for creating a file with no suffix when no other suffix rule applies is specified as an entry for the target rule `* : *`.

Implicit rules are inferred according to the suffixes listed as the dependencies of the built-in rule `.SUFFIXES`. Suffix order is significant; the first possible name for which both a file and a rule exists is inferred.

Built-in Rules

The following rules are special to `nmake`. Most are used to associate attributes with rules and targets. To avoid conflicts with these built-in rules, do not define uppercase rule names with a period (`.`) as the first character.

`.ARCHIVE` The dependencies of `.ARCHIVE` are suffixes associated with archive files (see `ar`(1) and `bld`(1)). A rule with this suffix is treated as an archive. When used as a dependency, `.ARCHIVE` causes the target to be treated as an archive and the `*.a : *` rule is used to update the target. The `$(ARUPDATE)` variable is set dynamically by `nmake` to contain the proper update sequence for the current archive target (the commands are preceded by a newline character). The default archive suffixes and update commands are as follows:

```
.ARCHIVE : .a

*.a : * (AR) (ARFLAGS)
        $(AR) $(ARFLAGS) $(<) $(>) $(ARUPDATE)
        $(RM) $(RMFLAGS) $(>)
```

`.ATTRIBUTE` When used as a dependency, `.ATTRIBUTE` defines the target as a user attribute. Dependencies of user attributes are treated as suffixes and a rule with one of these suffixes automatically inherits the corresponding user attribute. A rule can have any number of user attributes, up to an implementation-defined limit.

`.CLEAR` When used as a dependency, `.CLEAR` clears the dependency and the command lists for the target.

`.CURRENT` When used as a dependency, `.CURRENT` marks the target as being produced in the current directory. The dependencies of `.CURRENT` are suffixes associated with targets that are produced only in the current directory. The default is `.CURRENT : .a .o`.

`.DEFAULT` This rule is used as a last resort when no other rules can be inferred to make the current target.

`.DONE` This rule is made after all targets are made and has no effect on the update status of the targets. The commands are always executed in the foreground shell (see the Jobs subsection).

`.DONTCARE` When used as a dependency, `.DONTCARE` causes `nmake` to continue if the target cannot be made; otherwise, `nmake` issues an error and exits if a target cannot be made. This is the same functionality as the `-k` command-line option.

`.FOREGROUND` When used as a dependency, `.FOREGROUND` causes the target update commands to be executed in the foreground shell; otherwise, the commands can be executed in a background shell (see the Jobs subsection). Usually, `nmake` computes future dependencies while update commands are being executed; however, `.FOREGROUND` update commands cause `nmake` to block until the commands complete.

`.GLOBAL.`*x* The dependencies of `.GLOBAL.`*x* are inserted onto each target with suffix *.x* immediately before the target is made; *.x* may be the null suffix. For this rule, `$(<)` refers to the current `.GLOBAL.`*x* target, and `$(<<)` refers to the target on which the dependencies are inserted.

`.IMPLICIT` When used as a dependency, `.IMPLICIT` causes the implicit suffix rules to be applied even when update commands were specified for the target; otherwise, the implicit suffix rules are applied only to targets with no explicit update commands.

`.INIT` This rule is made before any other target and has no effect on the update status of the targets. The commands are always executed in the foreground shell (see the Jobs subsection).

`.INSERT` When used as a dependency, `.INSERT` causes the dependency list to be inserted rather than appended to the target dependency list.

`.INTERNAL` This rule is used internally.

`.INTERRUPT` This rule is made when an interrupt signal is caught; it has no effect on the update status of the targets. The commands are always executed in the foreground shell (see the Jobs subsection). `nmake` exits after the commands are executed.

`.MAIN` If no targets are explicitly listed on the command line, the dependencies of `.MAIN` are used as the main targets. If not explicitly specified in the input makefiles, the first dependency of `.MAIN` is set to be the first target encountered that is not a special rule or inference rule.

`.MAKE` When used as a dependency, `.MAKE` causes the command update list to be parsed by `nmake` rather than executed by the shell. Such command lists are always parsed, even with the `-n` option (the `-n` option traces but does not execute the target update command).

`.MAKEFILES` This rule specifies the default makefile names. If no explicit makefile is specified, the following files are tried in order:

```
.MAKEFILES : Nmakefile nmakefile Makefile makefile
```

`.MAKEINIT` This target is made before the `.SOURCE` targets are examined. `.INIT` is made after the `.SOURCE` targets are examined. Do not redefine this rule; however, it is safe to insert or append dependencies to `.MAKEINIT`. Variable assignments within `.MAKEINIT` commands override any command-line variable assignments.

`.NOEXPAND` This attribute is associated with state variables that are not to be expanded by the `:T=D:` edit operator. `.NOEXPAND` is defined using `.ATTRIBUTE`.

`.NOTOUCH` When used as a dependency, `.NOTOUCH` causes the target modify time to remain untouched, even if the corresponding update commands are executed. This allows initialization sequences to be specified for individual rules:

```
main : init header
        echo "executed if header is newer than main"
        init : .NOTOUCH
        echo "always executed for main"
```

`.NULL` When used as a dependency of a target with no suffix or explicit update commands, `.NULL` causes the commands that are associated with the `* : *` rule to be used when updating the target.

`.OPERATOR` When used as a dependency, `.OPERATOR` marks the target as an operator to be applied when makefiles are read. Operator names must consist of 2 characters.

`.OPTIONS` The dependencies of `.OPTIONS` are treated like command-line options. The options take effect immediately when `.OPTIONS` is read. The `-f` and `-g` options have no effect in this case. All options specified on the command line override the `.OPTIONS` specification, except for the `-j` option (which specifies the number of jobs that `nmake` can create concurrently).

`.PARAMETER` When used as a dependency, `.PARAMETER` marks the target as a parameter file that contains only definitions (for example, `#define` definitions) and comments. The modify time of a parameter file is ignored when determining the update status of corresponding targets.

`.POST` When used as a dependency, `.POST` causes the target to be made after the parent target is made. If a `.POST` dependency also has the `.NOTOUCH` attribute, commands are executed only if the target is updated.

`.PRECIOUS` If `nmake` is interrupted, the current target(s) are usually deleted; the dependencies of `.PRECIOUS`, however, are not deleted. If `.PRECIOUS` is specified with no dependencies, all targets are protected. Targets marked with `.ARCHIVE` are always precious.

`.PREFIXES` The dependencies of `.PREFIXES` are file name prefixes of the form *p*. They are used to infer implicit rules. The (left to right) prefix order is important; the first inference rule name for which both a file and a rule exist is inferred. The default prefix rules provide a smooth interface to Source Code Control System files (see `admin`(1)):

```
.PREFIXES : s.
```

`.READONLY` When used as a dependency for `.o` target files, `.READONLY` causes the data parts of the corresponding `.c` source file to be placed in read-only text. `.READONLY` can also be used to place the tables of corresponding `.l` and `.y` source files in read-only text. The commands are tailored to the current host machine. `.READONLY` is implemented as a `.USE` rule, and it will become obsolete when the `const` data attribute becomes a C language standard. Typical usage is as follows:

```
file.o : .READONLY
```

`.SEARCH` The dependencies of `.SEARCH` are suffixes associated with files that are to be searched for implicit file dependencies. When used as a dependency, `.SEARCH` marks the target to be searched for implicit file dependencies. By default, any dependency with a `.SEARCH` suffix or marked with `.SEARCH` is searched automatically for `#include` header file dependencies. If any files are newer than the dependency file, the corresponding target is updated. The default is as follows:

```
.SEARCH : .c .h .y .l .F
```

`.SOURCE` The dependencies of `.SOURCE` are directories to be scanned when searching for files. Because the directories are always searched in left-to-right order, the first directory that contains the file is used. The default is the current directory (.):

```
.SOURCE : .
```

`.SOURCE.`*x* The directories specified by the `.SOURCE.`*x* rule are checked for files with a period (.) as a suffix. If the file is not found, the directories specified by `.SOURCE` are checked. The *.x* suffix must be a dependency of `.SUFFIXES`. Because the directories are always searched in left-to-right order, the first directory that contains the file is used. The default when the `-X` option is disabled is as follows:

```
.SOURCE.a : /lib /usr/lib
.SOURCE.h : /usr/include
```

`.SUFFIXES` The dependencies of `.SUFFIXES` are file name suffixes of the form `.s`, used to infer implicit rules. The (left-to-right) suffix order is important; the first inference rule name for which both a file and a rule exist is inferred. The default suffix list is as follows:

```
SUFFIXES : .o .c .f .F .f90 .F90 .p  .y .l .s .sh .h .a .msg .exp .cat
```

The following list describes the meaning of each suffix:

Suffix	Meaning
`.o`	Relocatable
`.c`	C
`.f`	Fortran
`.F`	Fortran (processed)
`.f90`	CF90 Fortran

.F90	CF90 Fortran (processed)
.p	Pascal
.y	yacc(1)
.l	lex(1)
.s	Assembler
.sh	Shell script
.h	Header files
.a	Libraries
.msg	Message
.exp	Explanatory message
.cat	Catalog message

.TOUCH Each dependency of .TOUCH is touched as though it were already made. The dependencies are touched immediately when .TOUCH is read. This is the same functionality as the -t command-line option.

.UNTOUCH Each dependency of .UNTOUCH is untouched as though it were never made. The dependencies are untouched immediately when .UNTOUCH is read.

.USE When used as a dependency, .USE marks the target as a .USE rule. Any target that has a .USE rule as a dependency is updated using the commands associated with the .USE rule.

.WAIT .WAIT is a synonym for .FOREGROUND.

Command Execution

The nmake and sh(1) commands run as coprocesses. All update commands are sent to a single copy of the shell, keeping the shell environment intact between command executions (see the Jobs subsection for an exception). This includes the effects of cd(1) and shell parameter assignments. sh echoes each command when executed, unless the -s option is specified. Because entire command blocks are sent as a unit, special shell constructs (case, if, for, and while) can cross newline boundaries without newline escape characters indicating a continuation.

Commands returning nonzero status (see intro(1)) stop nmake, unless the -i or -k option is specified.

Interrupt and quit signals cause the current target to be deleted, unless the target is an archive, or a directory, or unless it is a dependency of the special rule .PRECIOUS.

Any command update list that contains the $(MAKE) variable is always executed, even when the -n option is specified. This simplifies the maintenance of a makefile hierarchy because one makefile can invoke other makefiles in the hierarchy, passing along the top-level options and variable assignments by using $(-) and $(=). You can use $(MAKE:) to disable execution when the -n option is specified.

Special Shell Commands

Some special shell commands are provided for the nmake and sh coprocess environment:

. . . The *ellipsis* command separates an update command list into precommands (before . . .) and postcommands (after . . .). The precommands are executed when a target is being updated. The postcommands are stacked (first-in, first-out) and are executed in the foreground shell after the last target is generated. The ellipses must appear as the first command on a line.

`exit` *code*
Removes the shell coprocess, stopping `nmake`.

`ignore` *shell-command*
Causes the exit status of *shell-command* to be ignored.

`_make_` *command data*
Passes messages from `sh`(1) to `nmake`. Do not modify the shell definition of either `_make_` or `make_id_`.

`_make_ error` *exit-code*
Called when an update command returns a nonzero exit code.

`_make_ exit`
Called when an update command block completes.

`_make_ start` *job-id process-id*
Associates a command block with a process ID.

`@ nmake-command`
The `nmake-command` is sent to and executed by `nmake`. This allows variable definitions and even makefiles to be modified dynamically (although some dynamic modifications may not work with the `-n` option). Embedded newline characters can be sent by enclosing the `nmake-command` in either single or double quotation marks.

`silent` *shell-command*
Prevents the shell from printing *shell-command.* If both `silent` and `ignore` are used, `silent` must precede `ignore`.

Jobs

The `-j` option allows `nmake` to update many targets concurrently. The updates are synchronized using the target dependency graph specified in `makefile`. With this option, each update command block is sent to a new subshell (background shell). Background shells inherit the environment of the main shell (foreground shell) and the foreground shell inherits the environment of `nmake`. You can force target update commands to execute in the foreground shell by including the special rule `.FOREGROUND` as a dependency.

Common Actions

When the `::` operator is used, several common action targets are defined automatically. The common action target *xxx* is defined as *.XXX* in the built-in rules. If *xxx* appears as a command-line target and *xxx* was not defined by the input makefiles, the target *.XXX* is made. The common action target must be the first command argument target (from left to right) and can be followed by specific command targets. If no command argument targets are specified, all command targets are affected. The common actions are as follows:

`arch` Creates an `ar`(1) archive of the source files that are listed after each `::` operator. The archive is placed in file *main*.arch; *main* is the base name of the main target rule.

`clean` Deletes all object files that correspond to the current makefile.

`clobber` Executes the `clean` action and also deletes the targets that correspond to the current makefile.

`cpio` Creates a `cpio`(1) archive of the source files listed after each `::` operator. The archive is placed in file *main*.`cpio`; *main* is the base name of the main target rule.

`ctags` Creates a *tags* file for `vi`(1) by using `ctags`(1).

`install` Makes the main target and copies it to the `$(INSTALLDIR)` directory. By default, `$(INSTALLDIR)` is `$(BINDIR)` for executable targets, `$(LIBDIR)` for object archive targets, and `$(MANDIR)`*n* for man page targets; *n* is the manual section number. `BINDIR`, `ETCDIR`, `INCLUDEDIR`, `LIBDIR`, `MANDIR`, and `NLSDIR` are defined as `$(ROOT)/bin`, `$(ROOT)/etc`, `$(ROOT)/include`, `$(ROOT)/man/man`, and `$(ROOT)/lib/nls/En`, respectively. The commands that are associated with the `.DOINSTALL` rule are used to do the copying. You can specify the installation directory for any (non-man page) target by using the `.INSTALL.`*x.* variable.

`lint` Runs `lint`(1) on the input source files. Any `.l` and `.y` source files are preprocessed automatically if necessary.

`lprof` Runs `lprof`(1) (systems other than Cray Research systems that support only System V) on the target command(s). Each command must have been generated using the `-ql` profiling option on the `lprof` command, and the *command*`.cnt` file must exist in the current directory (that is, *command* must have been run at least once).

`print` Prints the source files by passing them through the filter `$(PR)` and listing them with `$(LP)`.

`tar` Creates a `tar`(1) archive of the source files listed after each `::` operator. The archive is placed in file *main*.`tar`; *main* is the base name of the main target rule.

`uarch` Same as `arch` except only the source files modified since the last `uarch` are archived (see also `$(UTIME)` in `ucpio`).

`ucpio` Same as `cpio` except only those source files modified since the last `ucpio` are archived. If `$(UTIME)` is specified, it is assumed to be a file name whose modify time is used to determine which files are to be archived; only files that are newer than this modify time are archived.

`uprint` Same as `print` except only the source files modified since the last `uprint` are printed (see also `$(UTIME)` in `ucpio`). The `-F` option (forcing all targets to be updated) must be set the first time `uprint` is used.

`utar` Same as `tar` except that only the source files modified since the last `utar` are archived (see also `$(UTIME)` in `ucpio`).

NOTES

The default makefile ordering is the exact opposite of make(1). If old makefiles are named makefile and new makefiles are named Nmakefile, nmakefile, or Makefile, both make and nmake can be run without specifying explicit files names.

Because the built-in rules are placed in an nmake object file, performance is not degraded when different built-in rules are specified either by the MAKERULES environment variable or by using the -b and -g options.

The following command line causes all :: targets to be made by default:

```
.MAIN : .CLEAR .ALL
```

Otherwise, only the first :: target is made by default.

The :, :=, +=, operators, and operator lines are expanded when the makefile is read. Any $(*var*) variables that occur in these lines are frozen into the corresponding nmake object file. If the value of a frozen variable changes from one invocation of nmake to the next, either in a command-line definition or in the environment, a warning is issued and the makefile is recompiled automatically. You can defer variable evaluation on these lines by entering $(*var*) as $$(*var*).

It is not possible to specify explicit makefile dependencies; however, implicit makefile dependencies are generated automatically.

Some commands return inappropriate nonzero status; use the ignore command to avoid this.

The nmake utility detects only source files that exist before nmake is executed.

For a given command-name variable $(XX), the default flags are specified by the $(XXFLAGS) variable. For example, the flags for $(CC) are $(CCFLAGS), and the flags for $(YACC) are $(YACCFLAGS). You can use the $(LDLIBRARIES) variable to specify additional libraries to the cc(1) command. You can use the $(LINTLIBRARIES) variable to specify additional libraries to the lint(1) command. The $(LINTLIB) variable specifies the default lint(1) library directory.

The default yacc(1) rules for the *x*.y file produces the *x*.h and *x*.o files, not *y.tab*.h and *y.tab*.o or *y.tab*.c. The *x*.h file is updated only if it differs from the *x*.h file that is generated by the previous yacc. Similarly, the default lex(1) rules for the *x*.l file produce the *x*.o file, not lex.yy.o or *lex.yy*.c. Make the yacc target before any other targets that may depend on the generated header file.

For dynamic dependencies, #include <file> is treated as #include "file".

The nmake -l*x* library expansions can load different libraries than they would if the -l*x* options were passed directly to ld(1).

Because of optimizations, unified input syntax, and new functionality, this version of nmake is not compatible with either the original or the augmented version.

WARNINGS

A warning is issued when the input makefiles must be recompiled. A warning is issued if a rule can be bound to more than one file (unless the `-u` option is set); `nmake` proceeds with the first file found, using the `.SOURCE` and `.SOURCE`.*s* rules.

ENVIRONMENT VARIABLES

The `NPROC` environment variable specifies the number of jobs that `nmake` can execute concurrently. The `ARRANGED` variable specifies the location of the `arrange` daemon script. The built-in rules are compiled into an `nmake` object file whose location is specified by the `MAKERULES` environment variable. `MAKEPP` specifies the path name of the makefile preprocessor. `MAKESHELL` specifies the path name of the shell that is used to update targets. `SRCPATH` is interpreted as a colon-separated list of directories inserted in the `.SOURCE` and `.SOURCE.h` rule dependency lists (the current directory, designated by the period, is always searched first).

Do not redefine the following variables in the `nmake` object file; redefinitions of these environment variables can cause inconsistent system performance:

```
ARRANGED_FILE
ARRANGED_TMOUT
CPPINCLUDE
MAKE
MAKEFILE
ROOT
SHELL
```

BUGS

Syntactically incorrect `sh`(1) commands can cause `nmake` to hang; usually, it is caused by nonterminated strings enclosed in quotation marks.

The `nmake` utility is optimized to work with `sh`(1); it does not work with `csh`(1).

FILES

`Nmakefile, nmakefile, Makefile, makefile`	Default makefiles, which are tried in order
`/usr/lib/make/Makerules.mo`	Standard compiled built-in rules
`/usr/lib/make/cpp`	The `nmake` preprocessor
`/usr/lib/make/arranged`	Daemon used to sort output from one or more `nmake` files
`base.mo`	The `nmake` object file
`base.ms`	The `nmake` state file

SEE ALSO

admin(1), ar(1), cd(1), cpio(1), csh(1), ctags(1), ld(1), lex(1), lint(1), make(1), sed(1), tar(1), vi(1), yacc(1)

cc(1), cpp(1) in the *Cray Standard C Reference Manual*, Cray Research publication SR–2074

UNICOS Configuration Administrator's Guide, Cray Research publication SG–2303

NAME

nohup – Invokes a utility immune to hangups and quits

SYNOPSIS

nohup *utility* [*arguments*]

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The nohup utility executes *utility*, ignoring hangups and quits. If you do not redirect the output, it will be appended to nohup.out. If nohup.out cannot be created or opened for appending in the current directory, output is appended to the end of file $HOME/nohup.out.

If neither file can be created or opened for appending, *utility* is not invoked.

If the standard error is a terminal, all output written by the named *utility* to its standard error is redirected to the same file descriptor as the standard output.

The exit status of nohup is that of the utility specified by the *utility* operand.

NOTES

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
system, secadm	In a privileged administrator shell environment, shell-redirected I/O is not subject to file protections.
sysadm	Shell-redirected output is subject to security label restrictions.

If the PRIV_SU configuration option is enabled, shell-redirected I/O on behalf of the super user is not subject to file protections.

The csh(1) utility has a built-in nohup utility with slightly different characteristics (see csh(1)).

EXIT STATUS

The nohup utility exits with one of the following values:

126 The *utility* specified was found but could not be invoked.

127 An error occurred in the nohup utility or the *utility* specified could not be found.

Otherwise, the exit status of nohup is that of the utility specified by the *utility* operand.

EXAMPLES

This example puts the cc(1) compilation of example.c in the background, with the output going to nohup.out. User input is shown in courier bold:

```
$ nohup cc example.c &
10793
$ Sending output to nohup.out
```

FILES

nohup.out	File that contains output from *utility*
$HOME/nohup.out	File that contains output from *utility*

SEE ALSO

nice(1)

signal(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

NAME

nroff – Text formatting language

SYNOPSIS

nroff [-o*list*] [-n*N*] [-s*N*] [-r*aN*] [-i] [-q] [-z] [-m*name*] [-T*tty_type*] [-e] [-h] [-un] [*file_name(s)*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The nroff utility formats text contained in *file_name*(*s*) (standard input by default) for printing on typewriter-like devices and line printers.

An argument consisting of a dash (-) is taken to be a file name corresponding to the standard input. Options may appear in any order, but they must appear before the *file_name*(*s*) argument.

The nroff utility accepts the following options and operands:

-o*list* Prints only pages whose page numbers appear in the *list* of numbers and ranges separated by commas. A range *N*-*M* means pages *N* through *M*; an initial -*N* means from the beginning to page *N*; and a final *N*- means from *N* to the end. (See the BUGS section.)

-n*N* The first page generated begins with the number *N*.

-s*N* Stops every *N* pages. nroff halts *after* every *N* pages (default *N*=1) to allow paper loading or changing, and it resumes on receipt of a line feed or new line (new lines do not work in pipelines, for example, with mm(1)). This option does not work if the output of nroff is piped through col(1). When nroff halts between pages, an ASCII BEL is sent to the terminal.

-r*aN* Sets register *a* to *N*. This must have a 1-character name.

-i Reads standard input after *files* are exhausted.

-q Invokes the simultaneous input-output mode of the .rd request.

-z Prints only messages generated by .tm (terminal message) requests.

-m*name* Prepends to the input *file_name*(*s*) the macro file /usr/lib/tmac/tmac.*name*.

-T*tty_type* Prepares output for specified terminal. Known *tty_types* are as follows:

2631	Hewlett-Packard 2631 printer in regular mode
2631-c	Hewlett-Packard 2631 printer in compressed mode
2631-e	Hewlett-Packard 2631 printer in expanded mode
300	DASI-300 printer

300-12	DASI-300 terminal set to 12-pitch (12 characters per inch)
300s	DASI-300s printer (300S is a synonym)
300s-12	DASI-300s printer set to 12-pitch (12 characters per inch) (300S-12 is a synonym)
37	TELETYPE O Model 37 terminal (default)
382	DTC-382
4000a	Trendata 4000a terminal (4000A is a synonym)
450	DASI-450 (Diablo Hyterm) printer
450-12	DASI-450 terminal set to 12-pitch (12 characters per inch)
832	Anderson Jacobson 832 terminal
8510	C.ITOH printer
`lp`	Generic name for printers that can underline and tab (all text that use reverse line feeds, such as those having tables, that is sent to `lp` must be processed with `col`.)
tn300	GE Terminet 300 terminal
X	Printers equipped with TX print train

`-e` Produces equally spaced words in adjusted lines by using the full resolution of the particular terminal.

`-h` Uses output tabs during horizontal spacing to speed output and reduces output character count. Tab settings are assumed to be every 8 nominal character widths.

`-un` Sets the emboldening factor (number of character overstrikes) for the third font position (bold) to *n*, or to 0 if *n* is missing.

file_name(s) File that contains the text to be formatted.

BUGS

`nroff` uses Eastern Standard Time; therefore, depending on the time of the year and on your local time zone, the date that `nroff` generates may be off by one day from the date you think it is.

When `nroff` is used with the `-o`*list* option inside a pipeline (for example, with one or more `eqn`(1) and `tbl`(1) commands), it may cause a harmless "broken pipe" diagnostic if the last page of the document is not specified in *list*.

FILES

`/usr/lib/tmac/tmac.*`	Pointers to standard macro files
`/usr/lib/macros/*`	Standard macro files
`/usr/lib/nterm/*`	Terminal driving tables for `nroff`
`/usr/pub/terminals`	List of supported terminals

SEE ALSO

col(1), eqn(1), mm(1), tbl(1)

mm(7D) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

"The Formatter nroff: a Tutorial" in the *UNIX System V Documenter's Workbench Software User's Guide* (Prentice Hall, ISBN 0-13-943598-0)

"nroff/troff: Technical Discussion" in the *UNIX System V Documenter's Workbench Software Technical Discussion and Reference Manual* (Prentice Hall, ISBN 0-13-943580-8)

NAME

nslookup – Queries name servers interactively

SYNOPSIS

```
/usr/ucb/nslookup [-keyword[=value]]
/usr/ucb/nslookup [-keyword[=value]] host-to-find
/usr/ucb/nslookup [-keyword[=value]] -
/usr/ucb/nslookup [-keyword[=value]] - server address
/usr/ucb/nslookup [-keyword[=value]] - server name
```

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The nslookup utility queries DARPA Internet domain name servers. nslookup can operate in interactive and noninteractive mode. Interactive mode lets users query the name server for information about various hosts and domains or print a list of hosts in the domain. Noninteractive mode is used to print just the name and Internet address of a host or domain.

Interactive mode is entered in the following cases:

- When no arguments are specified (the default name server is used).
- When the first argument is a hyphen (-) and the second argument is the host name of a name server.

Noninteractive mode is used when the name of the host to be looked up is specified as the first argument. The optional second argument specifies a name server.

You can also use the command line to set status information that affects the lookups by preceding the status information keyword by a hyphen (-). See the set command in the next subsection for a complete list of keywords and descriptions.

Interactive Commands

You can interrupt commands at any time by typing your interrupt character (usually <CONTROL-c>). To exit, press <CONTROL-d> (EOF). The command line must consist of fewer than 80 characters. An unrecognized command is interpreted as a host name.

host [*server*] Looks up information for *host* by using the current default server or using *server* if it is specified.

server *domain*
lserver *domain*
Changes the default server to *domain*. lserver uses the initial server to look up information about *domain*; server uses the current default server. When an authoritative answer cannot be found, the names of servers that might have the answer are returned.

`root` Changes the default server to the server for the root of the domain name space. Currently, the `ns.nic.ddn.mil` host is used. (This command is a synonym for the `lserver ns.nic.ddn.mil` command.) The name of the root server can be changed using the `set root` command.

`finger` [*name*] [> *filename*]
`finger` [*name*] [>> *filename*]
Connects with the finger server on the current host. The current host is defined when a previous lookup for a host was successful and it returned address information (see the `set querytype=A` command). The *name* argument is optional. The > and >> characters can be used to redirect output in the usual manner.

`ls` *domain* [> *filename*]
`ls` *domain* [>> *filename*]
`ls -a` *domain* [> *filename*]
`ls -a` *domain* [>> *filename*]
`ls -h` *domain* [> *filename*]
`ls -h` *domain* [>> *filename*]
`ls -m` *domain* [> *filename*]
`ls -m` *domain* [>> *filename*]
`ls -s` *domain* [> *filename*]
`ls -s` *domain* [>> *filename*]
`ls -d` *domain* [> *filename*]
`ls -d` *domain* [>> *filename*]
`ls -t` [*type*] *domain* [> *filename*]
`ls -t` [*type*] *domain* [>> *filename*]
Lists the information available for *domain*. The default output contains host names and their Internet addresses. The `-a` option lists aliases of hosts in the domain. The `-h` option lists CPU and operating-system information for the domain. The `-m` option lists mail exchange information for the domain. The `-s` option lists well-known services for the domain. The `-d` option lists all contents of a zone transfer. The `-t` option lists all records of the specified type for the domain, or for the current `querytype` if *type* is not specified. When output is directed to a file, hash marks are printed for every 50 records received from the server.

`view` *filename* Sorts and lists the output of previous `ls` command(s) by using `more`(1).

`help`
`?` Prints a brief summary of commands.

`set` *keyword*[=*value*]
Changes status information that affects the lookups. Valid keywords are as follows:

`all"` Prints the current values of the various options to `set`. Information about the current default server and host is also printed.

`[no]debug` Turns on debugging mode. More information is printed about the packet sent to the server and the resulting answer. Turning debugging mode off by using `set nodebug` turns off all debugging, including that set by `set d2`. The default is `nodebug`; abbreviation is `[no]deb`.

`[no]d2` Turns on exhaustive debugging mode. Essentially, all fields of every packet are printed. Also turns on normal debugging mode, as if `set debug` had been issued. Turning off exhaustive debugging mode by using `set nod2`, however, leaves normal debugging mode on. The default is `nod2`.

`[no]defname` Appends the default domain name to every lookup. The default is `defname`; abbreviation is `[no]def`.

`[no]search` With `defname`, searches for each name in parent domains of the current domain. The default is `search`; abbreviation is `[no]sea`.

`domain=`*name* Changes the default domain name to *name*. If the `defname` option was set, the default domain name is appended to all lookup requests. The search list is set to parents of the domain with at least two components in their names. The default is value in host name or `/etc/resolv.conf`; abbreviation is `do`.

`querytype=`*value*
The default is `A`; abbreviation is `q`.

`type=`*value* Changes the type of information returned from a query to one of the following:

`A` 10	Host's Internet address (the default).
`CNAME` 10	Canonical name for an alias.
`HINFO` 10	Host CPU and operating system type.
`MB` 10	Mailbox domain name.
`MG` 10	Mail group member.
`MINFO` 10	Mailbox or mail list information.
`MR` 10	Mail rename domain name.
`MX` 10	The mail exchanger.
`NS` 10	Name server for the specified zone.
`PTR` 10	Pointer record for an Internet address in the `in-addr.arpa` domain.
`SOA` 10	Start Of Authority record for the named zone.
`WKS` 10	Well Known Services for the name.

The default is A; abbreviation is q or ty.

[no]recurse Instructs the name server to query other servers if it does not have the information. The default is recurse; abbreviation is [no]rec.

retry=*number* Sets the number of retries to *number*. When a reply to a request is not received within a certain amount of time (changed with set timeout), the request is resent. The retry value controls the number of times a request is resent before it gives up. The default is 4; abbreviation is ret.

root=*host* Changes the name of the root server to *host*. This affects the root command. The default is ns.nic.ddn.mil; abbreviation is ro.

timeout=*number*
Changes the time-out interval for waiting for a reply to *number* seconds. The default is 5; abbreviation is t.

[no]vc Specifies a virtual circuit for sending requests to the server. The default is novc; abbreviation is [no]v.

class=*value* Changes the current query class to one of the following:

IN 10	Internet records
CHAOS 10	CHAOSnet records
HESIOD 10	Hesiod records
ANY 10	Any record class

The default is IN; abbreviation is cl.

port=*number* Changes the number of the port used to connect the name server to *number*. The default is 53; abbreviation is po.

MESSAGES

When the look-up request is not successful, an error message is printed. Possible errors are as follows:

Time-out The server did not respond to a request after a certain amount of time (changed with set timeout=*value*) and a certain number of retries (changed with set retry=*value*).

No information Depending on the query type set with the set querytype command, information about the host was not available, although the host name is valid.

Nonexistent domain
The host or domain name does not exist.

Connection refused

`Network is unreachable`
: The connection to the name or finger server could not be made at the current time. This error commonly occurs with `finger` requests.

`Server failure`
: The name server found an internal inconsistency in its database and could not return a valid answer.

`Refused`
: The name server refused to service the request.

The following error indicates a bug in the program:

`Format error`
: The name server found that the request packet was not in the proper format.

FILES

`/etc/resolv.conf` Initial domain name and name server addresses

SEE ALSO

`more`(1)

RFC 1034, *Domain Names--Concepts and Facilities*, Mockapetris, November 1987
RFC 1035, *Domain Names--Implementation and Specification*, Mockapetris, November 1987

NAME

obc – Invokes the arithmetic language preprocessor

SYNOPSIS

obc [-c] [-l] [*files*]

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX
AT&T extensions (-c option)

DESCRIPTION

The obc utility (formerly, the UNICOS 8.0 version bc(1) utility) is an interactive processor for a language that resembles C but provides unlimited arbitrary-precision arithmetic. It takes input from any files specified and then reads from standard input.

The obc utility is actually a preprocessor for the odc(1) desk calculator utility, which obc invokes automatically unless you specify the -c (compile only) option. In this case, the odc(1) input is sent to standard output instead.

The obc utility accepts the following options and operand:

-c Specifies compile only. The output is sent to the standard output.

-l Defines the math functions and initializes scale to 20, rather than the default, which is 0.

files When using obc, assignment to scale influences the number of digits to be retained on arithmetic operations in the manner of odc(1).

You may use a letter simultaneously as an array, a function, and a simple variable. All variables are global to the program. auto variables are pushed down during function calls. When arrays are used as function arguments or defined as automatic variables, empty brackets must follow the array name.

Syntax and Functions

Comment strings are enclosed in /* and */.

Simple variable names can be any single lowercase letter (designated in the following examples as *l*). Array element names are expressed as *l*[*expression*]. Other acceptable variables are the words ibase (used to set the input number radix), obase (used to set the output number radix), and scale (used to set the number of digits to the right of the decimal).

Operands can be arbitrarily long numbers with optional sign and decimal points. Acceptable forms include (*expression*), `sqrt`(*expression*), `length`(*expression*), `scale`(*expression*), and *l*(*expression*). Both `scale` and `length` may be followed by a number that indicates the number of significant decimal digits.

Operators include `+`, `-`, `*`, `/`, `^`, and `%`; `%` is the remainder, and `^` is the exponent.

The `++` and `--` operators are used for prefix and postfix notation, and they are applied to variables.

Assignment operators include `=+`, `=-`, `=*`, `=/`, `=%`, and `=^`. These are used to change a named variable according to the operator. For example, the following two statements are equivalent:

```
s=+3
s=s+3
```

Statements are of the form *expression* or [*statement* `;` `...` `;` *statement*]. Either a semicolon or `<newline>` character can separate statements. Other statements include the following:

`if (`*expression*`)` *statement*
`while (`*expression*`)` *statement*
`for (`*expression* `;` *expression* `;` *expression*`)` *statement*
null statement
`break`
`quit`

Function definitions are of the following form:

```
define l (l, ... ,l) {
          auto ll ,..., l
          statement ; ... statement
          return (expression)
       }
```

Functions in the `-l` math library include the following:

`s`(*x*)	Sine
`c`(*x*)	Cosine
`e`(*x*)	Exponential
`l`(*x*)	Log
`a`(*x*)	Arctangent
`j`(*n*,*x*)	Bessel function

All function arguments are passed by value.

NOTES

The `&&` and `||` operators are unavailable.

A `for` statement must have all three *expression*s.

The `quit` statement is interpreted when read, not when executed.

EXIT STATUS

The `obc` utility exits with one of the following values:

0 All input files were processed successfully.

>0 An error occurred.

EXAMPLES

Example 1: The following is a simple example of `obc` functionality.

```
$ obc
78*67 + 20*58 + 55*69
10181
120*(67+20+55+69)
25320
p=(211*120)
q=10181
p-q
15139
quit
```

Example 2: The following example is input to `obc`, which defines a function that computes an approximate value of the exponential function:

```
scale = 20
define e(x){
        auto a, b, c, i, s
        a = 1
        b = 1
        s = 1
        for(i=1; 1==1; i++){
                a = a*x
                b = b*i
                c = a/b
                if(c == 0) return(s)
                s = s+c
                }
        }
```

Example 3: The following example is input to `obc`, which prints approximate values of the exponential function of the first 10 integers:

```
for(i=1; i<=10; i++) e(i)
```

FILES

`/usr/lib/lib.b` Mathematical library

SEE ALSO

`odc`(1)

NAME

od – Dump files in various formats

SYNOPSIS

od [-A *address_base*] [-j *skip*] [-N *count*] [-t *type_string*] [-v] [*file...*]

Obsolescent version; may not be supported in future releases:

od [-b] [-c] [-d] [-f] [-n *nl*] [-o] [-s] [-x] [-p] [-B] [-C] [-W] [*file*] [[+]*offset*[.][b]]

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The od utility dumps *file* in one or more formats as selected by the argument. The *file* operand specifies the file(s) to be dumped. If no file argument is specified, the standard input is used. If no options are specified, -t o2 is the default.

The od utility accepts the following options:

-A *address_base*
: Specifies the input offset base. The *address_base* argument is a character. The characters d, o, and x specify that the offset base is written in decimal, octal, or hexadecimal, respectively. The character n specifies that the offset is not written.

-j *skip*
: Jumps over *skip* bytes from the beginning of the input. od seeks past the first *skip* bytes in the concatenated input files. If the combined input does not consist of at least *skip* bytes, od writes a diagnostic message to standard error and exits with a nonzero exit status.

  By default, the *skip* is interpreted as a decimal number. With a leading 0x or 0X, *skip* is interpreted as a hexadecimal number; otherwise, with a leading 0 it is interpreted as an octal number. When you append the character b, k, or m to *offset*, it is interpreted as a multiple of 512, 1024, or 1048576 bytes, respectively.

-N *count*
: Formats no more than *count* bytes of input. By default, *count* is interpreted as a decimal number. With a leading 0x or 0X, *count* is interpreted as a hexadecimal number; otherwise, with a leading 0, it is interpreted as an octal number. If *count* bytes of input (after successfully skipping, if -j is specified) are not available, input that is available is formatted.

`-t` *type_string*

Specifies one or more output types. The *type_string* argument is a string that specifies the types to be used when writing the input data. The string may consist of the type specification characters `a`, `c`, `d`, `f`, `o`, `u`, and `x`, specifying named character, character, signed decimal, floating point, octal, unsigned decimal, and hexadecimal, respectively. The type specification characters `d`, `f`, `o`, `u`, and `x` can be followed by an optional unsigned decimal integer that specifies the number of bytes to be transformed by each instance of the output type. The type specification character `f` can be followed by an optional `F`, `D`, or `L` indicating that the conversion should be applied to an item of type *float*, *double*, or *long double*, respectively. The type specification characters `d`, `o`, `u`, and `x` can be followed by an optional `C`, `S`, `I`, or `L` indicating that the conversion should be applied to an item of type *char*, *short*, *int*, or *long*, respectively. Multiple types can be concatenated within the same *type_string* and multiple `-t` options can be specified. Output lines are written for each type specified in the order in which the type specification characters were given.

For characters and integers, byte sizes of 1, 2, 4, and 8 are supported. For floating-point data types, byte size of 8 and 16 are supported.

`-v` Writes all input data. Without this option, any number of groups of output lines, which would be identical to the immediately preceding group of output lines (except for the byte offsets), are replaced with a line that contains only an asterisk (`*`).

In the obsolescent version, you may specify only one file operand. If no file argument is specified, the standard input is used.

The obsolescent version of `od` accepts the following options:

`-b` Interprets bytes in octal.

`-c` Interprets bytes in ASCII. Certain nongraphic characters appear as C escapes: null=`\0`, backspace=`\b`, `<form-feed>`=`\f`, `<newline>`=`\n`, `<carriage-return>`=`\r`, and `<tab>`=`\t`; others appear as 3-digit octal numbers.

`-d` Interprets words in unsigned decimal (equivalent to `-t u2`).

`-f` Interprets words in floating-point format.

`-n` *nl* Prints *nl* lines, two words per line.

`-o` Interprets words in octal (equivalent to `-t o2`).

`-s` Interprets words in signed decimal (equivalent to `-t d2`).

`-x` Interprets words in hexadecimal (equivalent to `-t x2`).

`-p` Prints parcels rather than words.

`-B` Prints address in bytes.

`-C` If the format is that of `-d`, `-o`, `-s`, or `-x`, prints an additional column that contains the ASCII interpretation of the bytes. A period represents nonprintable characters.

`-W` Prints address in words (default for all except `-b` and `-c`).

offset Specifies the offset in the file where dumping will commence. Usually, this argument is interpreted as octal bytes. If `.` is appended to *offset*, the offset is interpreted in decimal. If *offset* begins with `0x`, the base is hexadecimal. The base generates the address column on the left; thus `+0.` prints decimal addresses. If `b` is appended to *offset*, the offset is interpreted in blocks of 4096 bytes. If the *file* argument is omitted, the *offset* argument must be preceded by `+`.

Dumping continues until the end-of-file is reached, or until *nl* lines have been printed.

NOTES

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
`system`, `secadm`	Allowed to dump any file.
`sysadm`	Allowed to dump any file, subject to security label restrictions. Shell-redirected I/O is subject to security label restrictions.

If the `PRIV_SU` configuration option is enabled, the super user is allowed to dump any file.

The input data is manipulated in blocks of 16 bytes. Each input block is written as transformed by each output type, one per written line, in the order that the output types were specified.

When the specified character type specification is chosen, all nongraphic characters are written as a three-character name.

EXIT STATUS

The `od` utility exits with one of the following values:

0 All input files were processed successfully.

>0 An error occurred.

EXAMPLES

Example 1: The following example assumes that you know that the file `myfile` contains ASCII characters. It is formatted as 1-byte character data shown on one line, and 8-byte hexadecimal data shown on another line. For easier study and review of the dump, the `od` output is redirected to a file named `mylist`:

```
od -t cx myfile > mylist
```

Example 2: The following example does the same task as Example 1, but it produces slightly different output by using the obsolescent version of `od`:

```
od -Cx myfile > mylist
```

Example 3: The following example dumps `512` bytes of the `textfile` file starting at byte `100`, suppressing the printing of the offset:

```
od -j 100 -N 512 -A n textfile
```

SEE ALSO

General UNICOS System Administration, Cray Research publication SG–2301

NAME

odc – Invokes the desk calculator

SYNOPSIS

odc [*file*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The odc utility (formerly, the UNICOS 8.0 version dc(1) utility) is an arbitrary-precision arithmetic package. Ordinarily it operates on decimal integers, but you may specify an input base, output base, and several fractional digits to be maintained. (See obc(1).) The overall structure of odc is a stacking (reverse Polish) calculator. When an argument is specified, input is taken from *file* until its end and is then taken from standard input. The following constructions are recognized:

number Pushes the value of *number* onto the stack. *number* is an unbroken string of the digits 0 through 9. It may be preceded by an underscore (_) to input a negative number. Numbers may contain decimal points.

file Optional file containing input.

+ - * / % ^
Adds (+), subtracts (-), multiplies (*), divides (/), remainders (%), or exponentiates (^) the top two values on the stack. The two entries are popped off the stack; the result is pushed onto the stack in their places. Any fractional part of an exponent is ignored.

s *x* Pops the value off the top of the stack and stores it in a register named *x*, which can be any character. When the s is capitalized, *x* is treated as a stack and the value is pushed onto it.

l *x* Pushes the value in register *x* onto the stack. Register *x* is not altered. All registers start with 0 value. When the l is capitalized, register *x* is treated as a stack and its top value is popped off and pushed onto the main stack.

c Pops all values off the stack.

d Duplicates the top value on the stack.

f Prints all values of the stack.

i Pops the top value off the stack and uses it as the number radix for further input.

I Pushes the input base onto the top of the stack.

k Pops the top value off the stack and uses that value as a nonnegative scale factor. The appropriate number of decimal places is printed on output and maintained during multiplication, division, and exponentiation. The interaction of scale factor, input base, and output base will be reasonable if all are changed together.

K	Pushes the value of the scale factor onto the stack.
o	Pops the top value off the stack and uses it as the number radix for further output.
O	Pushes the output base onto the top of the stack.
p	Prints the top value of the stack. The top value remains unchanged.
P	Interprets the top of the stack as an ASCII string, removes it, and prints it.
Q	Pops the top value off the stack. The string execution level is popped by that value.
q	Exits the program. When a string is executed, the recursion level is popped by two.
v	Replaces the top element on the stack with its square root. Any existing fractional part of the argument is taken into account, but otherwise the scale factor is ignored.
x	Treats the top element of the stack as a character string and executes it as a string of odc commands.
X	Replaces the number on the top of the stack with its scale factor.
z	Pushes the value of the stack level onto the stack.
Z	Replaces the number on the top of the stack with its length.
[*string*]	Puts the bracketed ASCII string on the top of the stack.
<*x* >*x* =*x* !<*x* !>*x* !=*x*	Pops and compares the top two elements of the stack. Register *x* is evaluated if the values obey the stated relation. The exclamation point indicates negation.
!	Interprets the rest of the line as a UNICOS command. Control returns to odc when the command terminates.
?	Takes a line of input from the input source (usually the terminal) and executes it.
; :	Used by obc(1) for array operations.

MESSAGES

x is unimplemented	The *x* is an octal number.
stack empty	Not enough elements on the stack to do what was requested.
Out of space	Free list is exhausted (too many digits).
Out of headers	Too many numbers are kept.
Out of pushdown	Too many items on the stack.
Nesting depth	Too many levels of nested execution.

EXAMPLES

Example 1: The following is a simple example of odc functionality.

```
$ odc
210
100+
98+
63*
p
25704
quit
```

Example 2: The following example prints the first 10 values of n!:

```
$ odc
[la1+dsa*pla10>y]sy
0sa1
lyx
1
2
6
24
120
720
5040
40320
362880
3628800
quit
```

SEE ALSO

obc(1)

NAME

pack, pcat, unpack – Compresses and expands files

SYNOPSIS

```
pack [-] [-f] files
pcat files
unpack files
```

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The pack utility tries to store the specified files in a compressed form. Wherever possible (and useful), each input file *file* is replaced by a packed file *file*.z with the same access modes, access and modified dates, and owner as those of *file*. The -f option is useful for causing an entire directory to be packed even if some of the files will not benefit. If *file*.z is successfully created, *file* will be removed. Packed files can be restored to their original form by using unpack or pcat.

The pack utility uses Huffman (minimum redundancy) codes on a byte-by-byte basis.

The pack utility accepts the following options:

- If you use the - argument, an internal flag is set, which prints the number of times each byte is used, its relative frequency, and the code for the byte on the standard output. Additional occurrences of - in place of *file* cause the internal flag to be set and reset.

-f Forces the packing of *files*.

files Specifies the files to be compressed or expanded.

The amount of compression obtained depends on the size of the input file and the character frequency distribution. Because a decoding tree forms the first part of each .z file, it is usually not worthwhile to pack files smaller than three blocks, unless the character frequency distribution is very skewed, which may occur with printer plots or pictures.

Typically, text files are reduced to 60% to 75% of their original size. Load modules, which use a larger character set and have a more uniform distribution of characters, show little compression, the packed versions being about 90% of the original size.

The pack utility returns a value representing the number of files that it failed to compress.

No packing occurs if any one of the following is true:

- The file appears to be already packed.
- The file has links.
- The file is a directory.
- The file cannot be opened.
- No disk storage blocks are saved by packing.
- A file called *file*`.z` already exists.
- The `.z` file cannot be created.
- An I/O error occurred during processing.

Directories cannot be compressed. The `pcat` utility does for packed files what `cat`(1) does for ordinary files, except that `pcat` cannot be used as a filter. The specified files are unpacked and written to the standard output. Thus, to view a packed file named *file*`.z`, enter one of the following:

```
pcat file.z
pcat file
```

To make an unpacked copy, `nnn`, of a packed file named `file.z` (without destroying `nnn`), use the following command:

```
pcat file >nnn
```

The `pcat` utility returns the number of files it was unable to unpack. If one or all of the following are true, failure may occur.

- The file cannot be opened.
- The file does not appear to be the output of *pack*.

The `unpack` utility expands files created by `pack`. For each file specified in the command, a search is made for a file called *file*`.z` (or just *file*, if *file* ends in `.z`). If this file appears to be a packed file, it is replaced by its expanded version. The new file has the `.z` suffix removed from its name, and it has the same access modes, access and modification dates, and owner as those of the packed file.

The `unpack` utility returns a value that is the number of files it was unable to unpack. Failure may occur for the same reasons that it may in `pcat`, as well as for the following reasons:

- A file with the ‘‘unpacked’’ name already exists.
- The unpacked file cannot be created.

NOTES

There are many zeros in Cray Research system executable files. Compression of 50% is possible, but the average is 35%.

SEE ALSO

`cat`(1), `compress`(1)

`chown`(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

NAME

passwd – Changes login password

SYNOPSIS

passwd [-b] [*name*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The passwd utility changes or installs a password associated with the login *name*.

Ordinary users may change only the password that corresponds to their login *name*.

The passwd utility prompts you for your old password if you have one. (If you did not have a previous password, the system skips the old-password prompt.) The system then provides a prompt for the new password. Because the system does not echo the password to the screen, it provides a second new-password prompt to verify the correctness of your first entry.

The UNICOS system can use machine-generated passwords (if RANDOM_PASS_ON is enabled). If this feature is enabled, you are not allowed to select your own password. Instead, a machine-generated password is presented to you when using the passwd utility (see the EXAMPLES section). You have the choice of accepting the password or requesting another. You may continue to request new passwords by entering a carriage return until a suitable password is presented. Normal password requirement checking is not done when this feature is enabled.

The passwd utility accepts the following option:

-b Allows batch users to change their password. This option cannot be used if the machine-generated password feature is enabled. The old password followed by the new password is read from stdin.

The system administrator is responsible for defining the *aging* time limit for passwords. This means that if you have attempted to use your old password beyond the length of time defined by the system administrator, you may not create a new password. In this case, the system rejects your new password attempt and the passwd utility terminates.

If the old password meets the aging requirements defined by the system administrator, the system checks to ensure that the new password meets the construction requirements that follow. If it does, and if the new password is entered correctly for the first and second prompt, the new password is accepted. (When the new password is entered a second time, the two copies of the new password are compared. If the two copies are not identical, the system repeats the prompting cycle.)

Passwords must meet the following requirements (these requirements are relevant only when the machine-generated password feature is not enabled):

- Each password must have at least 6 characters. Only the first 8 characters are significant.

- Each password must contain at least 2 alphabetic characters and at least 1 numeric or special character. In this case, ‘‘alphabetic’’ means mixed-case letters.
- Each password must differ from the user’s login *name* and any reverse or circular shift of that login *name*. For comparison purposes, an uppercase letter and its corresponding lowercase letter are equivalent.
- The new password must differ from the old by at least 3 characters. For comparison purposes, an uppercase letter and its corresponding lowercase letter are equivalent.

One whose effective user ID is 0 is called a super user; see `id`(1) and `su`(1). Because super users may change any password, `passwd` does not prompt super users for the old password. Super users are not forced to comply with password aging and password construction requirements. A super user can create a null password by entering a carriage return in response to the prompt for a new password.
NOTE: This is not recommended.

Only a user with an authorized `secadm` category may change another user’s password, according to the rules previously specified for the super user. Additionally, users are not allowed to log in with a null password.

NOTES

If this utility is installed with the default privilege assignment list (PAL), a user who is assigned the following privilege text upon execution of this command is allowed to perform the actions shown:

Privilege Text	Action
`chgany`	Allowed to change any password.

EXAMPLES

Example 1: The following example shows how to change your password if you are a batch user:

```
$ passwd -b username << EOF
oldpassword
newpassword
EOF
```

Example 2: The following example shows how to change your password when the machine-generated password feature is enabled:

```
$ passwd
Old password:
Your new password is:  lempamdo
Re-enter new password or (CR) to get another:
```

FILES

`/etc/udb` User validation file containing user control limits

SEE ALSO

`id`(1), `login`(1), `privtext`(1), `su`(1)

`crypt`(3C) in *UNICOS System Libraries Reference Manual*, Cray Research publication SR–2080

`udb`(5) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

General UNICOS System Administration, Cray Research publication SG–2301

NAME

paste – Merges same lines of files or subsequent lines of a file

SYNOPSIS

```
paste files ...
paste -d list files ...
paste -s [-d list] files ...
```

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The paste utility concatenates corresponding lines of the given input *files*. It treats each file as a column or columns of a table and pastes them together horizontally (parallel merging). It can be considered the counterpart of cat(1), which concatenates vertically (that is, one file after the other). In the last form in the SYNOPSIS section, paste combines subsequent lines of the input file (serial merging). In all cases, lines are glued together by using the <tab> character, or by using characters from an optionally specified *list*. Output is to standard output, so that it can be used as the start of a pipe, or as a filter, if - is used in place of a file name.

The paste utility accepts the following options:

-d Replaces the <newline> characters of each file except the last file (or last line in case of the -s option) with a <tab> character. This option allows replacing the <tab> character by one or more alternative characters (see below).

-s Merges subsequent lines rather than one from each input file. Use <tab> for concatenation, unless *list* is specified with the -d option. Regardless of the *list*, the last character of each file is forced to be a <newline> character.

list Replaces the default <tab> with one or more characters immediately following -d as the line concatenation character. The list is used circularly; that is, when exhausted, it is reused. In parallel merging (that is, no -s option), the lines from the last file are always terminated with a <newline> character, not from *list*. The list may contain the special escape sequences: \n (<newline>), \t (<tab>), \\ (backslash), and \0 (empty string, not a null character). If characters have special meaning to the shell, quoting may be necessary (for example, to get one backslash, use -d"\\\\").

- May be used in place of any file name to read a line from standard input. (No prompting occurs.)

files Specifies files in which lines will be merged.

NOTES

The `pr -t -m ...` command works similarly, but creates extra <blank>s, <tab>s, and <newline>s for a nice page layout.

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
`system`, `secadm`	Allowed to manage any input file. In a privileged administrator shell environment, shell-redirected I/O is not subject to file protections.
`sysadm`	Allowed to manage any input file subject to security label restrictions. Shell-redirected I/O is subject to security label restrictions.

If the `PRIV_SU` configuration option is enabled, the super user is allowed to manage any input file. Shell-redirected I/O on behalf of the super user is not subject to file protections.

EXIT STATUS

The `paste` utility exits with one of the following values:

0 Successful completion.

>0 An error occurred.

MESSAGES

`paste: too many files (limit 12)`
When the `-s` option is not specified, you can specify up to 12 input files.

`paste: cannot allocate enough memory`
When processing files with line lengths greater than 2048 bytes, `paste` tries to allocate more space for its buffers. If a memory limit is reached, this error message will appear.

EXAMPLES

Example 1: This example lists files in the directory in one column:

```
ls | paste -d" " -
```

Example 2: This example lists files in the directory in four columns:

```
ls | paste - - - -
```

Example 3: This example combines pairs of lines into lines:

```
paste -s -d"\t\n" file
```

SEE ALSO

cat(1), cut(1), grep(1), pr(1)

NAME

patch – Applies a diff(1) file to an original file

SYNOPSIS

patch [-c | -e | -n] [-b] [-d *dir*] [-D *define*] [-i *patchfile*] [-l] [-N] [-o *outfile*] [-p *number*] [-r *rejectfile*] [-R] *file*

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The patch utility takes a patch file that contains any of the three forms of difference listings produced by the diff(1) utility and applies those differences to an original file, producing a patched version. By default, the patched version is put in place of the original. This POSIX version produces numbered backup (of the form *.n*). The -b option puts the back-up files in a file that has the same name but with the extension .orig.

To specify where you want to place output, use the -o option; if that file already exists, it is backed up first.

If you omit the -i option, the patch is read from standard input. At startup, patch tries to determine the type of the differences (diffs) listing, unless overruled by a -c, -e, or -n option. Context diffs (old-style and new-style) and normal diffs are applied by the patch program itself, while ed(1) diffs are simply fed to the ed(1) utility through a pipe.

The patch utility tries to skip any leading lines, apply the diff, and then skip any trailing lines. Thus, you can feed an article or message containing a diff listing to patch, and it should work. If the entire diff is indented by a consistent amount, this will be taken into account.

With context diffs, and to a lesser extent with normal diffs, patch detects when the line numbers mentioned in the patch are incorrect, and tries to find the correct place to apply each hunk of the patch. As a first guess, it takes the line number mentioned for the hunk, plus or minus any offset used in applying the previous hunk. If that is not the correct place, patch scans both forward and backward for a set of lines that matches the context given in the hunk.

First, patch looks for a place at which all lines of the context match. If no such place is found, and it is a context diff and the maximum fuzz factor is set to 1 or more, another scan occurs ignoring the first and last line of context. If that fails, and the maximum fuzz factor is set to 2 or more, the first two and last two lines of context are ignored, and another scan is made. (The default maximum fuzz factor is 2.) If patch does not find a place to install that hunk of the patch, it puts the hunk out to a reject file, which is the name of the output file plus the .rej extension. (The rejected hunk will come out in context-diff form whether the input patch was a context-diff or a normal diff.) If the input was a normal diff, many of the contexts will be null.) The line numbers on the hunks in the reject file may differ from the patch file: they reflect the approximate location patch thinks the failed hunks belong in the new file, rather than the old one.

As each hunk is completed, you will be informed whether the hunk succeeded or failed, and the line on which (in the new file) patch thinks the hunk should go. If this differs from the line number specified in the diff, the offset is displayed. A single large offset may indicate that a hunk was installed in the wrong place. patch also indicates whether a fuzz factor was used to make the match.

If you do not specify an original file on the command line, patch tries to determine from the leading lines, the name of the file to edit. In the header of a context diff, the file name is found from lines that begin with *** or ---, with the shortest name of an existing file being used. Only context-diffs have lines like that but if an Index: line is in the leading lines, patch tries to use the file name from that line. The context-diff header takes precedence over an Index line. If no file name can be determined from the leading lines, you will be prompted for the file name to patch.

For example, while in a news interface, you can specify the following line and patch a file in the blurfl directory directly from the article that contains the patch:

```
| patch -d /usr/src/local/blurfl
```

If the patch file contains more than one patch, patch tries to apply each of them as if they came from separate patch files. This means that it is assumed that the name of the file to patch must be determined for each diff listing, and that the lines before each diff listing will be examined for items such as file names and revision level, as mentioned previously.

The patch utility recognizes the following options:

-b	Saves a backup file of the original contents of each modified file, before the differences are applied, in a file of the same name with the .orig extension appended to it.
-c	Interprets the patch file as a context diff (the output of the diff(1) utility when the -c or -C option is specified.)
-d *dir*	Changes the current directory to *dir* before processing.
-D *define*	The #ifdef. . .#endif constructs are used to mark changes. The *define* argument is used as the differentiating symbol.
-e	Interprets the patch file as an ed(1) script.
-i *patchfile*	Reads the patch information from the file named by the path name *patchfile*, rather than the standard input.

`-l` Causes any sequence of blank characters in the diff script to match any sequence of blank characters in the input file. Other characters are matched exactly.

`-n` Interprets the patch file as a normal diff.

`-N` Ignores patches that may be reversed or already applied. By default, already-applied patches are rejected; see the `-R` option.

`-o` *file* Writes a copy of the file referenced by each patch, with the appropriate differences applied, to *outfile*. Multiple patches for a single file are applied to the intermediate versions of the file created by any previous patches and result in multiple, concatenated versions of the file being written to *outfile*.

`-p` *number* Deletes *number* path name components from the beginning of each path name for all path names in the patch file that indicate the names of files to be patched. If the path name in the patch file is absolute, the leading slash(es) are considered the first component (for example, `-p 1` removes the leading slashes). Specifying `-p 0` causes the full path name to be used. If you omit the `-p` option, only the base name (the final path name component) is used.

`-r` *rejectfile* Overrides the default reject file name. In the default case, the reject file has the same name as the output file, with the suffix `.rej` appended to it.

`-R` Reverses the sense of the patch script. For example, assume that the diff script was created from the new version to the old version. The `-R` option cannot be used with `ed`(1) scripts. The `patch` utility attempts to reverse each portion of the script before applying it. Rejected differences is saved in swapped format. If this option is not specified, and until a portion of the patch file is successfully applied, `patch` attempts to apply each portion in its "reversed" sense as well as in its normal sense. If the attempt is successful, the user is prompted to determine if the `-R` option should be set.

ENVIRONMENT VARIABLES

`TMPDIR` Directory in which to put temporary files; default is `/tmp`.

NOTES

If you will be sending out patches, you should keep a `patchlevel.h` file, which is patched to increment the patch level as the first diff in the patch file you send out. Make sure you always specify the file names correctly, either in a context diff header, or with an `Index` line. If you are patching something in a subdirectory, tell the patch user to specify a `-p` patch as needed. You can create a file by sending out a diff that compares a null file to the file you want to create. This works only when the file you want to create does not exist already in the target directory. Do not send out reversed patches, because people may wonder whether they already applied the patch. Group related patches into separate easier-to-review files in case of problems.

EXIT STATUS

The `patch` utility exits with one of the following values:

0 Successful completion.

1 One or more lines were written to a reject file.

>1 An error occurred.

MESSAGES

The `Hmm...` message indicates that there is unprocessed text in the patch file and that `patch` is trying to see whether a patch is in that text and, if so, what kind of patch it is.

If any reject files were created, the `patch` utility exits with a nonzero status. When applying a set of patches in a loop, check this exit status so you don't apply a later patch to a partially patched file.

The `patch` utility cannot tell whether the line numbers are off in an `ed`(1) script, and can only detect bad line numbers in a normal diff when it finds a `change` or a `delete` command. A context diff using fuzz factor 3 may have the same problem. Until a suitable interactive interface is added, you should probably do a context diff in these cases to see if the changes made sense. Compiling without errors indicates that the patch probably worked, but not always.

The `patch` utility usually produces the correct results, even when it has to do a lot of guessing. However, the results are guaranteed to be correct only when the patch is applied to exactly the same version of the file from which the patch was generated.

BUGS

The `patch` utility could be smarter about partial matches, excessively-deviant offsets and swapped code, but that would take an extra pass. If code has been duplicated (for instance with `#ifdef OLDCODE ... #else ... #endif`), `patch` cannot patch both versions, and it may patch the wrong one and tell you that it succeeded.

If you apply a patch that you have already applied, `patch` treats it as a reversed patch and offers to unapply the patch.

FILES

`$TMPDIR/patch*`

SEE ALSO

`diff`(1), `ed`(1)

NAME

pathchk – Checks path names

SYNOPSIS

pathchk [-p] *pathnames...*

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The pathchk utility checks that one or more path names are valid (for example, they can be used to access or create a file without causing syntax errors) and portable (for example, no file name truncation will result). The -p option provides more extensive portability checks.

By default, the pathchk utility checks each component of each *pathname* operand based on the underlying file system. A diagnostic is written for each *pathname* operand that has one of the following characteristics:

- Is longer than {PATH_MAX} bytes
- Contains any component longer than {NAME_MAX} bytes in its containing directory
- Contains any component in a directory that cannot be searched
- Contains any character in any component that is not valid in its containing directory

It is not an error if one or more components of a *pathname* operand does not exist as long as a file matching the path name specified by the missing components can be created that does not violate any of the preceding checks.

The pathchk utility accepts the following option and operand:

-p Writes a diagnostic for each *pathname* operand rather than performing checks based on the underlying file system:

- Is longer than {_POSIX_PATH_MAX} bytes
- Contains any component longer than {_POSIX_NAME_MAX} bytes
- Contains any character in any component that is not in the portable file name character set

pathnames Denotes path names to be checked.

NOTES

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
`system`, `secadm`	In a privileged administrator shell environment, shell-redirected I/O is not subject to file protections.
`sysadm`	Shell-redirected output is subject to security label restrictions.

If the `PRIV_SU` configuration option is enabled, shell-redirected I/O on behalf of the super user is not subject to file protections.

EXIT STATUS

The `pathchk` utility exits with one of the following values:

0 All *pathname* operands passed all checks.

>0 An error occurred.

EXAMPLES

Example 1: The following shell script fragment verifies that all path names in an imported data interchange archive are legitimate and unambiguous on the current system:

```
pax -f archive | xargs pathchk
if [ $? -eq 0 ]
then
        pax -r -f archive
else
        echo Investigate problems before importing files.
        exit 1
fi
```

SEE ALSO

`getconf`(1)

`access`(2), `pathconf`(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

NAME

pax – Portable archive interchange

SYNOPSIS

pax [-cdnv] [-f *archive*] [-s *replstr*] ... [*pattern* ...]

pax -r [-cdiknuv] [-f *archive*] [-o *options*] ... [-p *string*] ... [-s *replstr*] ... [*pattern* ...]

pax -w [-dituvX] [-b *blocksize*] [-a] [-f *archive*] [-o *options*] ... [-s *replstr*] ... [-x *format*] [*files* ...]

pax -r -w [-diklntuvX] [-p *string*] ... [-s *replstr*] ... [*file* ...] *directory*

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The pax utility reads, writes, and lists members of archive files and copies directory hierarchies. Archive formats supported are those that conform to the Archive/Interchange File Format specified in *IEEE Std. 1003.1-1990* (POSIX.1). These archive formats include ustar and cpio.

The action to be taken depends on the presence of the -r and -w options. The four combinations of -r and -w are referred to as the four modes of operation: *list*, *read*, *write*, and *copy* modes, corresponding respectively to the four forms shown in the SYNOPSIS section.

Mode **Description**

list In *list* mode (when neither the -r option nor the -w option is specified), pax writes the names of the members of the archive file read from the standard input, with path names that match the specified patterns, to standard output. If a specified file is of type directory, the file hierarchy rooted at that file will also be written out. In *list* mode, if -f is not specified, the standard input will be an archive file.

In this mode, pax lists normal files one per line, hard link path names as:

pathname == *linkname*

and symbolic link path names as:

pathname -> *linkname*

The *pathname* argument is the name of the file being extracted, and *linkname* is the name of a file that appeared earlier in the archive. If *pathname* is a symbolic link file, *linkname* might not exist in the archive.

If the -v option is specified, pax lists normal path names in the same format used by ls(1) with the -l option. Hard links are shown as follows:

<ls -l listing> == *linkname*

Symbolic links are shown as follows:

<ls -l listing> -> *linkname*

read In *read* mode (when -r is specified, but -w is not), pax extracts the members of the archive file with path names that match the specified patterns. If an extracted file is of type directory, the file hierarchy rooted at that file will also be extracted. The extracted files are created relative to the current file hierarchy. In *read* mode, if -f is not specified, the standard input will be an archive file.

The extracted or copied output files are of the archived file type.

The ownership, access and modification times, and file mode of the restored files are discussed under the -p option.

write In *write* mode (when -w is specified, but -r is not), pax writes the contents of the *file* operands to the standard output in an archive format. A *file* operand of type directory includes all of the files in the file hierarchy rooted at the file. In *write* mode, the standard input is used only if no *file* operands are specified. Standard input will be a text file that contain a list of path names, one per line, without leading or trailing <blanks>. Symbolic links are not followed (that is, they are stored on the archive as symbolic links). If the -f option is specified, the archive will be written to the file specified by the argument.

copy In *copy* mode (when both -r and -w are specified), pax copies the *file* operands to the destination *directory*. If no *directory* operands are specified, a list of files to copy, one per line, is read from the standard input. A *file* operand of type directory includes all of the files in the file hierarchy rooted at the file.

The effect of the copy will be as if the copied files were written to an archive file and then subsequently extracted, except that there hard links may be between the original and the copied files.

WARNING: A recursive copy occurs if the destination directory is a subdirectory of one of the source files. It is an error for the file specified by the *directory* operand not to exist, not be writable by the user, or not be a file of type directory.

In *read* or *copy* modes, intermediate directories are created as necessary to preserve the proper file hierarchy of the archive members.

If any specified *pattern* or *file* operands are not matched by at least one file or archive member, pax writes a diagnostic message to standard error for each one that did not match and exits with a nonzero exit status.

The supported archive formats will be detected automatically on input. The default output archive format is the `ustar` format.

If the selected archive format supports the specification of hard-linked files, it is an error if these files cannot be linked when the archive is extracted.

The `pax` utility accepts the following options and operands:

`-r` Reads an archive file from standard input.

`-w` Writes files to the standard output in the specified archive format.

`-a` Appends files to the end of the archive. The `-w` and `-f` options must also be specified. Appending to character-special devices (for example, tapes) is not supported.

`-b` *blocksize* Blocks the output at a positive decimal integer number of bytes per write to the archive file. A `b` suffix multiplies *blocksize* by 512, a `k` suffix multiplies *blocksize* by 1024, and an `m` suffix multiplies *blocksize* by 1,048,576. Devices and archive formats may impose restrictions on blocking. Blocking is determined automatically in *read* mode. Default blocking when creating archives on disk is 32,256 bytes. Default blocking when creating archives on tape depends on the archive format. (See the `-x` option.) The largest *blocksize* allowed is 32,256 bytes.

`-c` Matches all file or archive members except those specified by the *pattern* or *file* operand.

`-d` Causes files of type directory being copied or archived or archive members of type directory being extracted to match only the file or archive member itself and not the file hierarchy rooted at the file.

`-f` *archive* Specifies the path name of the input or output archive, overriding the default standard input (in *list* or *read* mode) or standard output (*write* mode). The `/dev/tty` file is used to write prompts and read responses.

`-i` Interactively rename files or archive members. For each archive member that matches a *pattern* operand or file that matches a *file* operand, a prompt is written to file `/dev/tty`. A line is then read from `/dev/tty`. If this line is blank, the file or archive member will be skipped. If this line consists of one period, the file or archive member will be processed with no modification to its name; otherwise, its name is replaced with the contents of the line. If end-of-file is encountered when reading a response or if `/dev/tty` cannot be opened for reading and writing, the `pax` utility immediately exits with a nonzero exit status.

`-k` Prevents the overwriting of existing files.

`-l` Links files. In *copy* mode, hard links are made between the source and destination file hierarchies whenever possible.

`-n` Selects the first archive member that matches each *pattern* operand. No more than one archive member are matched for each pattern (although members of type directory will still match the file hierarchy rooted at that file).

`-o` *options* Provides information to `pax` to modify the algorithm for extracting or writing files that is specific to the file format specified by `-x`. This option currently does not provide any functionality.

`-p` *string* Specifies one or more file characteristic options (privileges). The *string* argument is a string that specifies file characteristics to be retained or discarded on extraction. The string consists of the specification characters `a`, `e`, `m`, `o`, and `p`. Multiple characteristics can be concatenated within the same string and multiple `-p` options can be specified. The meanings of the specification characters are as follows:

`a` Does not preserve file access times.

`e` Preserves the user ID, group ID, file mode bits, access time, and modification time.

`m` Does not preserve file modification times.

`o` Preserves the user ID and group ID.

`p` Preserves the file mode bits.

In the preceding list, preserve indicates that an attribute stored in the archive will be given to the extracted file, subject to the permissions of the invoking process; otherwise, the attribute will be determined as part of the normal file creation.

If neither the `e` nor the `o` specification character is specified, or the user ID and group ID are not preserved for any reason, `pax` does not set the `S_ISUID` and `S_ISGID` bits of the file mode.

If the preservation of any of these items fails for any reason, `pax` writes a diagnostic message to standard error. Failure to preserve these items affects the final exit status, but does not delete the extracted file.

If file-characteristic letters in any of the *string* option-arguments are duplicated or conflict with each other, the one(s) specified last will take precedence (for example, if you specify `-p eme`, file modification times will be preserved).

`-s` *replstr* Modifies file or archive member names specified by *pattern* or *file* operand according to the substitution expression *replstr*, using the syntax of the `ed`(1) utility. The concepts of *address* and *line* are meaningless in the context of the `pax` utility, and they are not supported. The format is as follows:

```
-s /old/new/[gp]
```

In `ed`(1), *old* is a basic regular expression; and *new* can contain an ampersand, *n* (*n* is a digit) backreferences, or subexpression matching. The *old* string also may contain `<newline>` characters.

Any nonnull character can be used as a delimiter (/ shown here). Multiple `-s` expressions can be specified; the expressions are applied in the order specified, terminating with the first successful substitution. The optional trailing `g` is defined in the `ed`(1) utility and specifies the replacement of every occurrence of *old* with the pattern *new*. The optional trailing `p` causes successful substitutions to be written to standard error. File or archive member names that substitute to the empty string are ignored when reading and writing archives.

If the `-s` option is specified, and the replacement string has a trailing `p`, substitutions will be written to standard error in the following format:

<original path name> `>>` *<new pathname>*

`-t` Causes the access times of the archived files to be the same as they were before being read by `pax`.

`-u` Ignores files that are older (having a less recent file modification time) than a preexisting file or archive member that has the same name. In *read* mode, an archive member that has the same name as a file in the file system will be extracted if the archive member is newer than the file. In *copy* mode, the file in the destination hierarchy is replaced by the file in the source hierarchy or by a link to the file in the source hierarchy if the file in the source hierarchy is newer.

`-v` In *list* mode, produces a verbose table of contents; otherwise, writes archive member path names to standard error.

If `-v` is specified in *read*, *write*, or *copy* mode, `pax` writes the path names it processes to the standard error.

`-x` *format* Specifies the output archive format. The `pax` utility recognizes the following formats:

`cpio` The extended `cpio` interchange format. The default *blocksize* for this format for character special archive files is 5120. All *blocksize* values less than or equal to 32,256 that are multiples of 512 are supported.

`ustar` The extended `tar` interchange format. The default *blocksize* for this format for character special archive files is 10,240. All *blocksize* values less than or equal to 32,256 that are multiples of 512 are supported.

Any attempt to append to an archive file in a format different from the existing archive format causes `pax` to exit immediately with a nonzero exit status.

`-X` When traversing the file hierarchy specified by a path name, `pax` will not descend into directories that have a different device ID (see `stat`(2)).

The options that operate on the names of files or archive members (`-c`, `-i`, `-n`, `-s`, `-u`, and `-v`) interact as follows.

In *read* mode, the archive members are selected based on the user-specified *pattern* operands as modified by the `-c`, `-n`, and `-u` options. Then, any `-s` and `-i` options will modify, in that order, the names of the selected files. The `-v` option writes names that result from these modifications.

In *write* mode, the files are selected based on the user-specified path names as modified by the `-n` and `-u` options. Then, any `-s` and `-i` options will, in that order, modify the names of these selected files. The `-v` option writes names that result from these modifications.

If both the `-u` and `-n` options are specified, `pax` will not consider a file selected unless it is newer than the file to which it is compared.

directory Specifies the destination directory path name for *copy* mode.

file Specifies a path name of a file to be copied or archived.

pattern Specifies a pattern that matches one or more path names of archive members. You may specify pattern in the name-generating notation of the pattern matching notation in `sh`(1). If no *pattern* is specified, all members in the archive are selected.

EXIT STATUS

The `pax` utility exits with one of the following values:

0 All files were processed successfully.

>0 An error occurred.

MESSAGES

If `pax` cannot create a file or a link when reading an archive or cannot find a file when writing an archive, or cannot preserve the user ID, group ID, or file mode when the `-p` option is specified, a diagnostic message will be written to standard error and a nonzero exit status will be returned, but processing will continue. In cases in which `pax` cannot create a link to a file, `pax` will not, by default, create a second copy of the file.

If the extraction of a file from an archive is terminated prematurely by a signal or error, `pax` may have only partially extracted the file or (if the `-n` option was not specified) may have extracted a file of the same name as that specified by the user, but which is not the file the user wanted. Additionally, the file modes of extracted directories may have additional bits from the `S_IRWXU` mask set, as well as incorrect modification and access times.

BUGS

Special permissions may be required to copy or extract files.

When getting an `ls -l` style listing on `ustar` format archives, link counts are listed as 0 because the `ustar` archive format does not keep link count information.

EXAMPLES

Example 1: The following command copies the file hierarchy rooted at the current directory to an archive file in the user's temporary directory:

```
pax -w -f $TMPDIR/archive .
```

Example 2: The following command copies the contents of `olddir` to `newdir`:

```
pax -rw olddir newdir
```

Example 3: The following command reads the archive file `pax.out` with all files rooted in `/usr` in the archive extracted relative to the current directory:

```
pax -r -s,/usr/,, -f pax.out
```

SEE ALSO

`ed`(1), `find`(1), `sh`(1), `ls`(1)

`chmod`(2), `stat`(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

ISO/IEC 9945-1:1990 (IEEE Std 1003.1-1990), *Information technology – Portable Operating System Interface (POSIX) – Part 1: System Application Program Interface (API)* [*C Language*]

NAME

perfscripts – Generates a file tree containing performance tool files

SYNOPSIS

perfscripts [-V]

IMPLEMENTATION

Cray PVP systems (except CRAY EL series)

DESCRIPTION

The perfscripts utility generates a file tree in your current directory. This tree contains several subtrees that contain scripts that can be used to generate reports from the raw output of the following performance tools:

- flowtrace
- perftrace
- prof(1)
- hpm(1)
- hpmall(8)

Each set of files has a README file, which explains how to use the scripts. In addition, sample data files are included that can be used with these scripts or with the appropriate post-processing tools such as flowview(1), perfview(1), and profview(1).

Each of these performance tools is discussed on its associated man page and in the *Guide to Parallel Vector Applications*, Cray Research publication SG–2182.

You are free to alter these scripts to your own specifications. However, note that Cray Research will support only the script contents as they are initially written in your directory by this utility.

The perfscripts utility executes a shell script to create the directories and write the files. If the script cannot perform these actions, it issues error messages. During normal script execution, you will see a number of messages that look like the following:

```
shar: Extracting "performance/perftrace/report" (2856 characters)
```

The perfscripts utility accepts the following option:

-V Lists version number of perfscripts, along with a copyright notice, before creating the directories for the performance tools.

EXAMPLES

The following example shows an execution of `perfscripts` using the `-V` option:

```
$ perfscripts -V
perfscripts VERSION 70.0

   (c) COPYRIGHT CRAY RESEARCH, INC.

    UNPUBLISHED -- ALL RIGHTS RESERVED UNDER
    THE COPYRIGHT LAWS OF THE UNITED STATES

shar: Creating directory "performance"
shar: Creating directory "performance/flowtrace"
 .....
 .....
 .....
```

SEE ALSO

`csh`(1), `hpm`(1), `prof`(1), `sh`(1)

`flowtrace`(7), `performance`(7), `perftrace`(7) (available only online)

`hpmall`(8) in the *UNICOS Administrator Commands Reference Manual*, Cray Research publication SR–2022

Guide to Parallel Vector Applications, Cray Research publication SG–2182

NAME

pg – File perusal filter for CRTs

SYNOPSIS

pg [-*number*] [-c] [-e] [-f] [-n] [-p *string*] [-s] [+*linenumber*] [+/*pattern*/] [*files*]

IMPLEMENTATION

All Cray Research systems

STANDARDS

XPG4

DESCRIPTION

The pg utility is a filter that allows the examination of each specified file one screenful at a time on a CRT. (The file name - and/or NULL arguments indicate that pg should read from the standard input.) Each screenful is followed by a prompt. If the user types a <carriage return>, another page is displayed; other possibilities are enumerated below.

This utility is different from previous paginators in that it allows you to back up and review something that has already passed. The method for doing this follows.

To determine terminal attributes, pg scans the terminfo(5) database for the terminal type specified by environment variable TERM. If TERM is not defined, the terminal type dumb is assumed.

The pg utility accepts the following options:

-*number* Specifies the number of lines the window pg is to use instead of the default. (On a terminal that has 24 lines, the default window size is 23.)

-c Homes the cursor and clears the screen before displaying each page. If clear_screen is not defined for this terminal type in the terminfo(5) database, this option is ignored.

-e Prevents pg from pausing at the end of each file.

-f Inhibits pg from splitting lines. Typically, pg splits lines longer than the screen width, but some sequences of characters in the text being displayed (for example, escape sequences for underlining) generate undesirable results.

-n This option causes an automatic end of command as soon as you enter a command letter. Typically, commands must be terminated by a <newline>.

-p *string* Causes pg to use *string* as the prompt. If the prompt string contains a %d, the first occurrence of %d in the prompt will be replaced by the current page number when the prompt is issued. The default prompt string is :.

-s Causes pg to print all messages and prompts in standout mode (usually inverse video).

+*linenumber* Starts at *linenumber*.

+/*pattern*/ Starts up at the first line containing the basic regular expression pattern.

files Files to be displayed.

The responses that may be typed when pg pauses can be divided into three categories: those causing further perusal, those that search, and those that modify the perusal environment.

Commands that cause further perusal usually take a preceding *address*, which is an optionally signed number indicating the point from which further text should be displayed. This address is interpreted in either pages or lines depending on the command. A signed address specifies a point relative to the current page or line, and an unsigned address specifies an address relative to the beginning of the file. Each command has a default address that is used if none is provided.

The perusal commands and their defaults are as follows:

(+1)<newline> or <blank>
Causes one page to be displayed. The address is specified in pages.

(+1) l With a relative address, causes pg to simulate scrolling the screen, forward or backward, the number of lines specified. With an absolute address, prints a screenful beginning at the specified line.

(+1) d or ^D Simulates scrolling half a screen forward or backward.

The following perusal commands take no address:

. or ^L Causes the current page of text to be redisplayed.

$ Displays the last windowful in the file. Use with caution when the input is a pipe.

The following commands are available for searching for text patterns in the text. The regular expressions described in ed(1) are available. They must always be terminated by a <newline>, even when the -n option is specified.

i/*pattern*/ Searches forward for the *i*th (default *i*=1) occurrence of *pattern*. Searching begins immediately after the current page and continues to the end of the current file, without wrap-around.

i^*pattern*
i?*pattern*? Searches backward for the *i*th (default *i*=1) occurrence of *pattern*. Searching begins immediately before the current page and continues to the beginning of the current file, without wrap-around. The ^ notation is useful for Adds 100 terminals that will not properly handle the ?.

After searching, pg usually displays the line found at the top of the screen. This can be modified by appending m or b to the search command to leave the line found in the middle or at the bottom of the window from now on. The t suffix can be used to restore the original situation.

The user of pg can modify the environment of perusal with the following commands:

*i*n	Begins perusing the *i*th next file in the command line. The *i* is an unsigned number; default value is 1.
*i*p	Begins perusing the *i*th previous file in the command line. The *i* is an unsigned number; default is 1.
*i*w	Displays another window of text. If *i* is present, sets the window size to *i*.
s *filename*	Saves the input in the named file. Only the current file being perused is saved. The white space between the s and *filename* is optional. This command must always be terminated by a <newline>, even when the -n option is specified.
h	Displays an abbreviated summary of available commands.
q or Q	Quits pg.
!*command*	*command* is passed to the shell, whose name is taken from the SHELL environment variable. If this is not available, the default shell is used. This command must always be terminated by a <newline>, even when the -n option is specified.

At any time when output is being sent to the terminal, you can press the quit key (<CONTROL-\>) or the interrupt (<CONTROL-c>) key. This causes pg to stop sending output and to display the prompt. The user may then enter one of the preceding commands in the normal manner. Unfortunately, some output is lost when this is done, because any characters waiting in the terminal's output queue are flushed when the QUIT signal occurs.

If the standard output is not a terminal, pg acts just like cat(1), except that a header is printed before each file (if there is more than one file).

NOTES

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
system, secadm	In a privileged administrator shell environment, shell-redirected I/O is not subject to file protections.
sysadm	Shell-redirected output is subject to security label restrictions.

If the PRIV_SU configuration option is enabled, shell-redirected I/O on behalf of the super user is not subject to file protections.

While waiting for terminal input, pg responds to <CONTROL-c>, <DEL>, and ^ by terminating execution. Between prompts, however, these signals interrupt the current task of pg and place the user in prompt mode. These should be used with caution when input is being read from a pipe, because an interrupt is likely to terminate the other commands in the pipeline.

Users of the more(1) utility will find that the z and f commands are available and that the terminal /, ^, or ? may be omitted from the searching commands.

EXIT STATUS

The pg utility exits with one of the following values:

0 Successful completion.

>0 An error occurred.

BUGS

If terminal tabs are not set every eight positions, undesirable results may occur.

When you use pg as a filter with another command that changes the terminal I/O options, terminal settings may not be restored correctly.

Nulls in a character string cause the current line to terminate on output.

EXAMPLES

The following is a sample usage of pg for reading system news:

```
news | pg -p "(Page %d):"
```

FILES

/usr/lib/terminfo/?/*	Terminal information database
/tmp/pg*	Temporary file when input is from a pipe

SEE ALSO

cat(1), ed(1), grep(1), more(1), pr(1)

terminfo(5) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

NAME

pr – Prints files

SYNOPSIS

pr [-*column*] [-w [*width*]] [-a] [+*page*] [-d] [-e[*ck*]] [-f] [-F] [-h *header*] [-i[*ck*]] [-l [*length*]] [-L] [-n[*ck*]] [-o [*offset*]] [-p] [-r] [-s[*separator*]] [-t] [*files*]

pr [-m] [-w [*width*]] [+*page*] [-d] [-e[*ck*]] [-f] [-F] [-h *header*] [-i[*ck*]] [-l [*length*]] [-L] [-n[*ck*]] [-o [*offset*]] [-p] [-r] [-s[*separator*]] [-t] [*files*]

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4
AT&T extensions (-f and -p options)

DESCRIPTION

The pr utility formats and prints the contents of a file. If *file* is -, or if no files are specified, pr assumes standard input. pr prints the specified files on standard output.

By default, the listing is separated into pages, each headed by the page number, a date and time, and the name of the file. (The date and time is the last modification time, unless input is redirected from standard input; in this case, the current date and time is displayed.) Page length is 66 lines, which includes 10 lines of header and trailer output. The header is composed of 2 blank lines, 1 line of text (which can be altered with -h) and 2 blank lines; the trailer is 5 blank lines. For single-column output, line width may not be set and is unlimited. For multicolumn output, line width may be set, and the default is 72 columns. Diagnostic reports (failed options) are reported at the end of standard output associated with a terminal, rather than interspersed in the output. Pages are separated by a series of <newline> characters rather than <form-feed> characters.

By default, columns are of equal width, separated by at least one <space>; lines that do not fit are truncated. If you specify the -s option, lines are not truncated and columns are separated by the *separator* character.

To produce multicolumn output, use either -*column* or -m; -a must be used only with -*column* and not -m.

The pr utility accepts the following options:

NOTE: Space is not permitted between the following options and their arguments, except for -h, -l, -o, and -w. The -h option requires a <space>.

-*column* Prints *column* columns of output (default is 1). Output appears as if -e and -i are turned on for multicolumn output. Do not use with -m.

`-m` Merges and prints all files simultaneously, one per column. The maximum number of files that may be specified is 8. If a line is too long to fit in a column, it is truncated. Do not use with `-`*column*.

`-w` [*width*] Sets the width of a line to *width* character positions (default is 72). This is effective only for multicolumn output (`-`*column* and `-m`). No line limit exists for single-column output.

`-a` Prints multicolumn output across the page, one line per column. *column* must be greater than 1. If a line is too long to fit in a column, it is truncated.

`+`*page* Begins printing with page numbered *page* (default is 1).

`-d` Double-spaces the output. Blank lines that result from double-spacing are dropped when they occur at the top of a page.

`-e`[*ck*] Expands each input `<tab>` character to character positions *k*+1, 2**k*+1, 3**k*+1, and so on. If *k* is 0 or is omitted, default tab settings at every eighth position are assumed. All `<tab>` characters in the input are expanded into the appropriate number of `<space>`s. If *c* (any nondigit character) is given, it is treated as the input `<tab>` character. (Default for *c* is the `<tab>` character.)

`-L` Folds the lines of the input file. When used in multicolumn mode (with the `-a` or `-m` option) lines will be folded to fit the current column's width; otherwise, they will be folded to fit the current line width.

`-F` Uses one `<form-feed>` character for new pages. (Default is to use a sequence of `<newline>` characters.)

`-f` Uses one `<form-feed>` character for new pages. (Default is to use a sequence of `<newline>` characters.) If the standard output is associated with a terminal, it pauses before beginning the first page.

`-h` *header* Uses *header* as the text line of the header to be printed instead of the file name. `-h` is ignored when `-t` is specified, or when `-l` *length* is specified and the value of *length* is 10 or fewer. A `<space>` is required between the `-h` option and its argument.

`-i`[*ck*] In output, replaces white space when possible by inserting `<tab>` characters to character positions *k*+1, 2**k*+1, 3**k*+1, and so on. If *k* is 0 or is omitted, default tab settings at every eighth position are assumed. If *c* (any nondigit character) is given, it is treated as the output `<tab>` character (default for *c* is the `<tab>` character).

`-l` [*length*] Sets the length of a page to *length* lines (default is 66). `-l0` is reset to `-l66`. When the value of *length* is 10 or less, `-t` appears to be in effect because headers and trailers are suppressed. By default, output contains 5 lines of header and 5 lines of trailer, leaving 56 lines for user-supplied text. If *length* exceeds 10, *length*–10 lines are left per page for user-supplied text. When *length* is 10 or less, header and trailer output is omitted to make room for user-supplied text.

`-n`[*ck*]	Provides *k*-digit line numbering (default for *k* is 5). The number occupies the first *k*+1 character positions of each column of single-column output or each line of `-m` output. If *c* (any nondigit character) is given, it is appended to the line number to separate it from whatever follows (default for *c* is a `<tab>`).
`-o` [*offset*]	Offsets each line by *offset* character positions (default is 0). The number of character positions per line is the sum of the width and offset.
`-p`	Pauses before beginning each page if the output is directed to a terminal. (`pr` rings the bell at the terminal and waits for a `<carriage-return>`.)
`-r`	Prints no diagnostic reports on files that will not open.
`-s`[*separator*]	Separates columns by the single-character *separator* instead of by the appropriate number of `<space>` characters. (Default for *separator* is a `<tab>`.) Prevents truncation of lines on multicolumn output, unless `-w` is specified.
`-t`	Does not print the five-line identifying header or the five-line trailer. Quits printing after the last line of each file without spacing to the end of the page. Use of the `-t` options overrides the `-h` option.
files	Specifies the files to print.

NOTES

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
`system`, `secadm`	In a privileged administrator shell environment, shell-redirected I/O is not subject to file protections.
`sysadm`	Shell-redirected output is subject to security label restrictions.

If the `PRIV_SU` configuration option is enabled, shell-redirected I/O on behalf of the super user is not subject to file protections.

EXIT STATUS

The `pr` utility exits with one of the following values:

0 All files were written successfully.

>0 An error occurred.

EXAMPLES

Example 1: To print `file1` and `file2` as a double-spaced, three-column listing headed by `file list`, enter the following command line:

```
$ pr -3dh "file list" file1 file2
```

Example 2: To copy `file1` to `file2`, expanding tabs to columns 10, 19, 28, 37, . . ., enter the following command line:

```
$ pr -e9 -t <file1 >file2
```

Example 3: To print `file1` and `file2` simultaneously in a two-column listing, with no header or trailer, where both columns have line numbers, enter the following command line:

```
$ pr -t -n file1 | pr -t -m -n file2 -
```

Example 4: To print `file1` with line numbers 0001 through 9999 and page breaks and headers every 66 lines, enter the following:

```
$ pr -n4 -l66 file1
```

FILES

`/dev/tty*` Terminal devices

SEE ALSO

`cat`(1), `more`(1), `pg`(1)

NAME

printenv – Prints the environment variable values

SYNOPSIS

/usr/ucb/printenv *name*

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The printenv utility prints the values of the variables in the environment.

The printenv utility accepts the following operand:

name When *name* is specified, only the value is printed.

EXIT STATUS

If you specify *name* and it is not defined in the environment, printenv returns exit status 1; otherwise, it returns status 0.

SEE ALSO

env(1), sh(1)

NAME

printf – Writes formatted output

SYNOPSIS

printf *format* [*argument* ...]

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The printf utility writes formatted operands to the standard output. The *argument* operands are formatted under control of the *format* operand.

The printf utility accepts the following operands:

format A string that describes the format to use to write the remaining operands.

argument The strings to be written to standard output, under the control of *format*.

The *format* operand is used as the *format* string described in printf(3C), except for the following:

- A <space> character in the format string, in any context other than a flag of a conversion specification, is treated as an ordinary character that is copied to the output.
- The following escape sequences are supported:

  \\ Represents the backslash character
  \a Represents the <alert> character (octal 07)
  \b Represents the <backspace> character (octal 010)
  \f Represents the <form-feed> character (octal 014)
  \n Represents the <newline> character (octal 012)
  \r Represents the <carriage-return> character (octal 015)
  \t Represents the <tab> character (octal 011)
  \v Represents the <vertical-tab> character (octal 013)

  In addition, *ddd* is supported; *ddd* is a one-, two-, or three-digit octal number that is written as a byte with the numeric value specified by the octal number.
- The printf utility will not precede or follows output from the d or u conversion specifications with <blank> characters not specified by the *format* operand.
- The printf utility will not precede output from the o conversion specification with zeros not specified by the *format* operand.

- An additional conversion character, `b`, is supported as follows. The argument is taken to be a string that may contain backslash-escape sequences. The following backslash-escape sequences are supported:
  - The escape sequences listed in item 2 are converted to the characters they represent.
  - `\0`*ddd*; *ddd* is a zero-, one-, two-, or three-digit octal number that is converted to a byte with the numeric value specified by the octal number.
  - `\c`, which is not written and causes `printf` to ignore any remaining characters in the string operand that contains it, any remaining string operands, and any additional characters in the *format* operand.

  The interpretation of a backslash followed by any other sequence of characters is interpreted as the character immediately following the backslash.

  Bytes from the converted string are written until the end of the string or the number of bytes indicated by the precision specification is reached. If the precision is omitted, it will be taken to be infinite, therefore, all bytes up to the end of the converted string are written.
- For each specification that consumes an argument, the next argument operand is evaluated and converted to the appropriate type for the conversion as specified below.
- The *format* operand is reused as often as necessary to satisfy the argument operands. Any extra `c` or `s` conversion specifications are evaluated as if a null string argument were supplied; other extra conversion specifications are evaluated as if a 0 argument were supplied. If the *format* operand contains no conversion specifications and *argument* operands are present, the *argument*s are ignored.
- If a character sequence in the *format* operand begins with a `%` character, but does not form a valid conversion specification, the character immediately following the `%` is printed.

The *argument* operands are treated as strings if the corresponding conversion character is `b`, `c`, or `s`; otherwise, it is evaluated as a C constant, as described by the C Standard, with the following extensions:

- A leading plus or minus sign is allowed.
- If the leading character is a single or double quotation mark, the value is the numeric value in the underlying code set of the character following the single or double quotation mark.

If an argument operand cannot be completely converted into an internal value appropriate to the corresponding conversion specification, a diagnostic message is written to standard error and the utility does not exit with a zero exit status, but it continues processing any remaining operands and writes to standard output the value accumulated at the time the error was detected.

NOTES

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
`system`, `secadm`	In an administrator privileged shell environment, shell-redirected I/O is not subject to security label restrictions.

If the PRIV_SU configuration option is enabled, and if the user is the super user, shell-redirected I/O is not subject to security label restrictions.

EXIT STATUS

The printf utility exits with one of the following values:

0 Successful completion.

>0 An error occurred.

EXAMPLES

Example 1: The following shell script fragment prompts for two responses from the user:

```
printf "\aPlease fill in the following: \nName: "
read name
printf "Phone Number: "
read phone
```

Example 2: The following example reads from a file that contains two integer values per line, right and wrong, then calculates the quotient, printing it out as follows:

```
$ while read right wrong ; do
>      percent=$(echo "scale=1;($right*100)/($right+$wrong)" | bc)
>      printf "%2d right\+%2d wrong\+(%s%%)\n" $right $wrong $percent
> done < $TMPDIR/database_file
```

Example 3: The following example produces the following output:

```
$ printf "%5d%4d\n" 1 21 321 4321 54321
    1  21
  3214321
54321   0
```

SEE ALSO

echo(1), sh(1)

printf(3C) in the *UNICOS System Libraries Reference Manual*, Cray Research publication SR–2080

NAME

`privtext` – Gets the privilege text of a file

SYNOPSIS

`privtext` [`-a`] [`-c` *category_list*] *files...*

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `privtext` utility displays the privilege text character sequence. The privilege text displayed is the one assigned to you when you invoke the files identified by *files*, based on your current active category.

If you specify the `-c` option, `privtext` displays the privilege text that is assigned to any user who has any of the active categories specified in *category_list*, upon execution of each of the files identified by *files*.

If the `-a` option is specified, *privtext* displays privilege text information for each of your authorized categories.

The `privtext` utility accepts the following options and operands:

`-a`	Displays privilege text for each of the user's authorized categories.
`-c`	Displays privilege text for each active category specified in *category_list*.
category_list	List of active categories. If multiple categories are specified, they must be separated by commas, with no intervening white space.
files	File names for which privilege text is to be retrieved.

NOTES

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
`system`, `secadm`	In a privileged administrator shell environment, shell-redirected I/O is not subject to file protections.
`sysadm`	Shell-redirected output is subject to security label restrictions.

If the `PRIV_SU` configuration option is enabled, shell-redirected I/O on behalf of the super user is not subject to file protections.

EXIT STATUS

The `privtext` utility exits with one of the following values:

0 Privilege text was successfully displayed for all specified commands.

1 An invalid or badly formed option was supplied.

2 Privilege text was displayed for some, but not all commands.

4 Privilege text could not be displayed for any of the specified commands.

EXAMPLES

The following examples assume that you have the `secadm` and `sysadm` categories authorized, the privilege assignment list (PAL) for `othercommand` is empty, and `somecommand` has been assigned the following PAL:

```
a:PRIV_NULL
f:PRIV_DAC_OVERRIDE,PRIV_MAC_READ,PRIV_MAC_WRITE
s:PRIV_DAC_OVERRIDE,PRIV_MAC_READ,PRIV_MAC_WRITE

system:PRIV_DAC_OVERRIDE,PRIV_MAC_READ,PRIV_MAC_WRITE:texta
secadm:PRIV_DAC_OVERRIDE,PRIV_MAC_READ,PRIV_MAC_WRITE:texta
sysadm:PRIV_DAC_OVERRRIDE:textb
other:PRIV_NULL:TEXT_NULL
```

Example 1: The following example shows the retrieval of the privilege text assigned to you if you have no active categories when you invoke `somecommand`:

```
$ privtext somecommand
other:TEXT_NULL
```

Example 2: The following example shows the use of the `-c` option, which displays the privilege text for the specified active category. In this example, the display shows the privilege text assigned to you if you have an active `sysadm` category when you invoke `somecommand`.

```
$ privtext -c sysadm somecommand
sysadm:textb
```

Example 3: The following example shows the use of the `-a` option, which displays privilege text assigned to all categories authorized for you when you invoke `somecommand`. The `other` category is retrieved to show the case where you do not have any active categories.

```
$ privtext -a somecommand
secadm:texta
sysadm:textb
other:TEXT_NULL
```

Example 4: The following example shows the use of the `-c` option, which displays the privilege text assigned to a user with any of the specified categories active. Use of this option allows you to determine what privilege text would be assigned to an administrative role, not a specific user. Note that you do not need to have the category authorized to retrieve information about the category. Because the PAL for `somecommand` does not have an entry for `netadm`, the match is made from the `other` entry:

```
$ privtext -c netadm,secadm somecommand
netadm:TEXT_NULL
secadm:texta
```

Example 5: The following example shows the use of both the `-a` and `-c` options, which display the privilege text for each category that is authorized for you and for any user with an active `netadm` category (even though the `netadm` category is not authorized for you). The `other` category is retrieved to show the case where you do not have any active categories:

```
$ privtext -a -c netadm,secadm somecommand
netadm:TEXT_NULL
secadm:texta
sysadm:textb
other:TEXT_NULL
```

Example 6: The following example shows the privilege text retrieved for `somecommand` and `othercommand` for each category authorized for you. The `other` category is retrieved to show the case where you do not have any active categories.

```
$ privtext -a somecommand othercommand
# somecommand:
secadm:texta
sysadm:textb
other:TEXT_NULL
# othercommand:
secadm:TEXT_NULL
sysadm:TEXT_NULL
other:TEXT_NULL
```

NAME

procstat – Gathers I/O and process statistics

SYNOPSIS

procstat [-R *rawfile*] [-r *rptfile*] [-m] [-i] [-d] [-V] *commands*

IMPLEMENTATION

Cray PVP systems

DESCRIPTION

The procstat utility produces information on the current activity of the specified commands. The information always includes process start and exit, and by default includes memory activity and I/O usage (including secondary data segments (SDS) usage). The recommended way to read procstat output is by means of the procview(1) utility; the procstat -r option can also be used to create a report with a format that is less convenient.

The procstat utility accepts the following options:

-R *rawfile* Writes file *rawfile* from which procview can create a report. This is the recommended usage. Either -r or -R (or both) must be specified.

-r *rptfile* Writes a formatted report to *rptfile*.

-m Reports on memory activity, as reported by sbreak (see brk(2)). This information is included by default; this option has the effect of turning off -i unless that is also specified.

-i Reports specifically on I/O activity (including SDS usage). This information is included by default; this option has the effect of turning off -m unless that is also specified.

-d Gathers detailed I/O information on each operation counted by procstat. This option should be used with caution, because it can generate a very large amount of information.

-V Lists the version of procstat on stderr, along with a copyright message.

commands Specifies the commands to be monitored by procstat. Arguments to these commands can be included on the same line. There is no default.

NOTES

The procstat utility does not support macrotasked programs. Autotasked programs are supported only if the NCPUS environment variable is set to 1. The statistics provided by procstat are limited to tracking by process ID, not by multitasking group. With multitasking groups, the statistics for file I/O need to be associated with OS multitasking groups to be correct.

Several more system calls, including ialloc(2), dup(2), fork(2), and ioctl(2) do not have hooks to provide information to procstat. Problems that stem from misuse of the heap may abort when run under procstat, even though they appear to execute correctly in normal circumstances.

I/O waiting times shown for interactive problems will be high for files such as stdin and stdout.

Statistics for READA and WRITEA may be inaccurate.

If a program aborts when run with procstat, it is usually caused by the program's incorrect use of the malloc(3C) subroutine.

BUGS

When a program terminates abnormally, some of the information still in the buffers in child processes is not flushed down the pipe.

The default report does not include all the information in the packets from the child processes.

Asynchronous I/O is not very well handled.

The procstat utility cannot identify certain file types, such as sockets.

EXAMPLES

The following example writes a report in file a.report, for command a.out, which has the arguments arg1 and arg2.

```
$ procstat -R a.raw a.out arg1 arg2
$ procview -L -Sn a.raw > a.report
```

SEE ALSO

procview(1)

brk(2), dup(2), fork(2), ialloc(2), ioctl(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

malloc(3C) in the *UNICOS System Libraries Reference Manual*, Cray Research publication SR–2080

performance(7) (available only online)

NAME

prs – Prints an SCCS file

SYNOPSIS

prs [-d[*dataspec*]] [-r[*SID*]] [-e] [-l] [-c *date-time*] [-a] *files*

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The prs utility prints, on the standard output, parts or all of a Source Code Control System (SCCS) file (see sccsfile(5)) in a user-supplied format. If a directory is named, it behaves as though each file in the directory were specified as a named file, except that non-SCCS files (last component of the path name does not begin with s.) and unreadable files are silently ignored.

Options to prs, which may appear in any order, consist of keyletter arguments and file names.

All the described options apply independently to each named file.

The prs utility accepts the following options:

-d[*dataspec*] Specifies the output data specification. The *dataspec* is a string consisting of SCCS file data keywords (see the Data Keywords subsection) interspersed with optional user-supplied text.

-r[*SID*] Specifies the SCCS identification (SID) string of a delta for which information is desired. If no SID is specified, the SID of the most recently created delta is assumed.

-e Requests information for all deltas created earlier than and including the delta designated, using the -r option or the date specified by the -c option.

-l Requests information for all deltas created later than and including the delta designated, using the -r option or the date specified by the -c option.

-c[*date-time*] Cutoff *date-time*. Appears in the form:

```
YY[MM[DD[HH[MM[SS]]]]]
```

Units omitted from the *date-time* default to their maximum possible values. That is, -c7502 is equivalent to -c750228235959. Any number of nonnumeric characters may separate the various 2-digit units of the *date-time* in the form: "-c77/2/2 9:22:25".

`-a` Requests printing of information for both removed deltas. That is, delta type = *R* (see `rmdel`(1)) and existing deltas, that is, delta type = *D*. If the `-a` option is not specified, information for only existing deltas is provided.

files Specifies the SCCS files to print.

Data Keywords

Data keywords (see table 1) specify which parts of an SCCS file are to be retrieved and output. All parts of an SCCS file (see `sccsfile`(5)) have an associated data keyword. There is no limit on the number of times a data keyword may appear in a *dataspec*.

The information printed by `prs` consists of: the user-supplied text and appropriate values (extracted from the SCCS file) substituted for the recognized data keywords in the order of appearance in the *dataspec*. The format of a data keyword value is either simple (S), in which keyword substitution is direct, or multiline (M), in which keyword substitution is followed by a carriage return.

User-supplied text is any text other than recognized data keywords. A tab is specified by `\t` and carriage return/`new line` is specified by `\n`. The default data keywords are as follows:

```
":Dt:\t:DL:\nMRs:\n:MR:COMMENTS:\n:C:"
```

Table 1. SCCS Files Data Keywords

Keyword	Data item	File section	Value	Format
`:Dt:`	Delta information	Delta table	See below†	S
`:DL:`	Delta line statistics	"	`:Li:/:Ld:/:Lu:`	S
`:Li:`	Lines inserted by delta	"	*nnnnn*	S
`:Ld:`	Lines deleted by delta	"	*nnnnn*	S
`:Lu:`	Lines unchanged by delta	"	*nnnnn*	S
`:DT:`	Delta type	"	`D`~or~`R`	S
`:I:`	SCCS ID string (SID)	"	`:R:.:L:.:B:.:S:`	S
`:R:`	Release number	"	*nnnn*	S
`:L:`	Level number	"	*nnnn*	S
`:B:`	Branch number	"	*nnnn*	S
`:S:`	Sequence number	"	*nnnn*	S
`:D:`	Date delta created	"	`:Dy:/:Dm:/:Dd:`	S
`:Dy:`	Year delta created	"	*nn*	S
`:Dm:`	Month delta created	"	*nn*	S
`:Dd:`	Day delta created	"	*nn*	S
`:T:`	Time delta created	"	`:Th:::Tm:::Ts:`	S
`:Th:`	Hour delta created	"	*nn*	S
`:Tm:`	Minutes delta created	"	*nn*	S
`:Ts:`	Seconds delta created	"	*nn*	S
`:P:`	Programmer who created delta	"	*logname*	S
`:DS:`	Delta sequence number	"	*nnnn*	S

Table 1. SCCS Files Data Keywords (continued)

Keyword	Data item	File section	Value	Format
:DP:	Predecessor delta seq #	"	*nnnn*	S
:DI:	Seq # of deltas included, excluded, or ignored	"	:Dn:/:Dx:/:Dg:	S
:Dn:	Deltas included (seq #)	"	:DS:~:DS:. . .	S
:Dx:	Deltas excluded (seq #)	"	:DS:~:DS:. . .	S
:Dg:	Deltas ignored (seq #)	"	:DS:~:DS:. . .	S
:MR:	MR numbers for delta	"	*text*	M
:C:	Comments for delta	"	*text*	M
:UN:	User names	User names	*text*	M
:FL:	Flag list	Flags	*text*	M
:Y:	Module type flag	"	*text*	S
:MF:	MR validation flag	"	yes~or~no	S
:MP:	MR validation program name	"	*text*	S
:KF:	Keyword error/warning flag	"	yes~or~no	S
:KV:	Keyword validation string	"	*text*	S
:BF:	Branch flag	"	yes~or~no	S
:J:	Joint edit flag	"	yes~or~no	S
:LK:	Locked releases	"	:R:. . .	S
:Q:	User-defined keyword	"	*text*	S
:M:	Module name	"	*text*	S
:FB:	Floor boundary	"	:R:	S
:CB:	Ceiling boundary	"	:R:	S
:Ds:	Default SID	"	:I:	S
:ND:	Null delta flag	"	yes~or~no	S
:FD:	File descriptive text	Comments	*text*	M
:BD:	Body	Body	*text*	M
:GB:	Gotten body	"	*text*	M
:W:	A form of what(1) string	NA	:Z::M:\t:I:	S
:A:	A form of what(1) string	NA	:Z::Y:~:M:~:I::Z:	S
:Z:	what(1) string delimiter	NA	@(#)	S
:F:	SCCS file name	NA	*text*	S
:PN:	SCCS file path name	NA	*text*	S

† :Dt:~=~:DT:~:I:~:D:~:T:~:P:~:DS:~:DP:

MESSAGES

Error messages from SCCS are printed. Use help(1) for explanations.

EXAMPLES

Example 1:

```
$ prs -d"Users and/or user IDs for :F: are: \n:UN:" s.file
```

May produce the following on the standard output:

```
Users and/or user IDs for s.file are:
xyz 131 abc
```

```
$ prs -d"Newest delta for pgm :M:: :I: Created :D: By :P:" -r s.file
```

May produce the following on the standard output:

```
Newest delta for pgm file: 1.2 Created 92/03/12 By cl
```

Example 2: As a special case:

```
$ prs s.file
```

May produce the following on the standard output for each delta table entry of the D type:

```
s.file:

D 1.2 92/03/12 09:56:16 cl 2 1  00001/00000/00005
MRs:
COMMENTS:
added one more line

D 1.1 92/03/12 09:45:26 cl 1 0  00005/00000/00000
MRs:
COMMENTS:
date and time created 92/03/12 09:45:26 by cl
```

The only keyletter argument allowed to be used with the special case is the -a option.

FILES

/tmp/pr*?????* Temporary working file

SEE ALSO

admin(1), cdc(1), comb(1), delta(1), get(1), help(1), rmdel(1), sact(1), sccsdiff(1), unget(1), val(1), vc(1), what(1)

sccsfile(5) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

NAME

ps – Reports process status

SYNOPSIS

ps [-a] [-A] [-c *corefile*] [-d] [-e] [-f] [-g *pgrplist*] [-G *grplist*] [-j *jidlist*] [-l] [-L] [-m] [-M] [-n *namelist*] [-o *format*] [-p *proclist*] [-t *termlist*] [-u *uidlist*] [-U *uidlist*] [-w]

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4
AT&T extensions (-c, -n, and -u options)
CRI extensions (-j, -m, -L, and -M options)

DESCRIPTION

The ps utility prints certain information about active processes. Without options, information is printed only about processes associated with the same effective user ID and the same controlling terminal as the invoker. The output is a short listing that consists of only the process ID, terminal identifier, cumulative time, and command name. Otherwise, the options control information that is displayed.

Options that use lists as arguments can have the list specified in one of two forms: a list of identifiers separated from one another by a comma, or a list of identifiers enclosed in double quotation marks and separated from one another by a comma and/or one or more spaces.

The ps utility accepts the following options:

-a	Prints information about all processes, except process group leaders and processes, not associated with a terminal.
-A	Same as -e option.
-c *corefile*	Uses the *corefile* file in place of /dev/mem.
-d	Prints information about all processes, except process group leaders.
-e	Prints information about all processes.
-f	Generates a full listing.
-g *pgrplist*	Restricts listing to data about processes that have process group leaders specified in *pgrplist*.
-G *grplist*	Restricts listing to data about processes whose group ID numbers or group names are given in *grplist*.
-j *jidlist*	Restricts listing to data about processes whose job ID numbers are specified in *jidlist*.
-l	Generates a long listing.

`-L` Same as `-l` option except that the PRI field is represented symbolically.

`-m` (CRAY T3D systems only) Prints information about active CRAY T3D processes and the partition with which they are associated. This option prints the UNICOS process information and the CRAY T3D partition information.

`-M` (CRAY T3D systems only) Prints only information about CRAY T3D partitions.

`-n` *namelist* Takes *namelist* as the name of an alternative system namelist file in place of `/unicos`. Only appropriately authorized users can successfully use this option.

`-o` *format* Produces a listing according to the specified *format*. (*format* is defined in the Format Specification subsection.) If you specify multiple `-o` options, the format specification will be interpreted as the space-separated concatenation of all *format* option arguments. The `-o` option takes precedence over the `-f`, `-l`, `-L`, and `-w` options.

`-p` *proclist* Restricts the listing to data about processes that have process ID numbers specified in *proclist*.

`-t` *termlist* Restricts the listing to data about the processes associated with the terminals specified in *termlist*. You may specify terminal identifiers in one of two forms: the device's file name (such as `tty02`) or, if the device's file name starts with `tty` or `ttyp`, just the identifier following the characters `tty` or `ttyp` (such as `02`, `004`, or `p004`).

`-u` *uidlist* Restricts the listing to data about processes that have user ID numbers or login names in *uidlist*. In the listing, the numerical user ID is printed, unless the `-f` option is used, in which case, the login name is printed.

`-U` *uidlist* Same as `-u` option.

`-w` Generates a wide listing.

The column headings and the meaning of the columns in a `ps` listing follows. The letters `f`, `w`, or `l` indicate the options (full, wide, or long, respectively) that cause the corresponding heading to appear; `all` means that the heading always appears. These options determine only the information provided for a process; they do not determine which processes will be listed.

`ACCTID` (`w`) Account ID of the process.

`ADDR` (`w,l`) Memory address of the process, if resident; otherwise, the disk address.

`C` (`w,l`) Processor utilization.

`CM` (`w`) CPU mask; used to limit a process to one or more specified CPUs.

`CMD` (`all`) Command name.

`CPUTIME` (`w`) Cumulative execution time for the process to a precision of hundredths of seconds.

`F` (`w,l`) Flags (octal) associated with the process. For more information on flags, see `proc.h` in the `/usr/include/sys` directory.

`HIMEM` (`w`) Maximum size (in blocks) that the process has grown.

`JID` (`w`) Job ID of the process.

NI (w,l)	Nice value; used in priority computation.
PID (all)	Process ID of the process; if you know this number and if you are the owner or root, you can kill the process.
PPID (f,w,l)	Process ID of the parent process.
PRI (w,l)	Priority of the process; higher numbers mean lower priority.
S (w,l)	The state of the process: I Intermediate R Running S Sleeping T Stopped W Waiting Z Terminated
SCTIME (w)	Cumulative system call time for the process.
STIME (w)	Starting time of the process. Not supported.
SWAPS (w)	Number of times the process has been swapped.
SZ (w,l)	Size in blocks of the core image of the process.
TIME (all except w)	Cumulative execution time for the process.
TTY (all)	Controlling terminal for the process.
UID (f,w,l)	User ID number of the process owner; the login name is printed under the -f option.
WAITSWAP (w)	Cumulative time that the process has been swapped.
WCHAN (w,l)	The event for which the process is waiting or sleeping; if blank, the process can be run.

When you specify the -m or -M option, you receive the following additional information:

ETIME (all)	Wall-clock execution time for the CRAY T3D process minutes and seconds in the following format: m:s
PES (m, M)	Number of processing elements allocated to the CRAY T3D process.
PRTN (all)	CRAY T3D partition ID of the process; this identifier is used for obtaining partition statistics from the mppstat(8) utility.
SHAPE (w, l)	CRAY T3D partition shape; (XxYxZ) hardware partition only.
STATE (all)	CRAY T3D partition state: Z ZOMBIE A ACTIVE

F FROZEN
E ERROR
U UNKNOWN

TYPE (all) CRAY T3D partition type; hardware or operating system.

When a process has exited and has a parent, but the parent has not waited for it, the process is marked <defunct>.

Format Specification

The -o option allows the output format to be specified under user control. The format specification consists of a list of field names presented as a single argument, separated by a blank or comma. Each field has a default header, which you can override by appending an equals sign and the new text of the header, such as -o **user=FOO**. The rest of the characters in the argument are used as the header text. The fields specified are arranged in columns and are written in the order specified on the command line. The width of a field will be at least as wide as the header text for that field. If the header text is null, such as -o "user=", the field will have no header text. If all header text fields are null, no header line is written.

The following field names are valid for the -o *format* argument:

Field name	Default header	Description
acctid	ACCTID	Account ID of the process.
address	ADDRESS	Memory address of the process if resident; otherwise, the disk address (octal).
addr	ADDRESS	Same as address.
args	COMMAND	Not supported. Required for POSIX and XPG4 compliance.
c	C	Processor utilization. Required for XPG4 compliance.
cmd	CMD	Command name.
comm	COMMAND	Same as cmd. Required for POSIX and XPG4 compliance.
command	COMMAND	Command name (longer version of command name).
cpumask	CPUMASK	CPU mask; used to limit a process to one or more specified CPUs (octal).
cm	CPUMASK	Same as cpumask.
cputime	CPUTIME	Cumulative execution time for the process.
etime	ELAPSED	Not supported. Required for POSIX and XPG4 compliance.
flags	FLAGS	Flags associated with the process (octal).
group	GROUP	Group name associated with the effective group ID of the process. Required for POSIX and XPG4 compliance.
himem	HIMEM	Maximum size in blocks that the process has grown to (decimal).
jid	JID	Job ID of the process.
nice	NI	Nice value; used in priority computation. Required for POSIX and XPG4 compliance.
ni	NI	Same as nice.

Field name	Default header	Description
pcpu	%CPU	Not supported. Required for POSIX and XPG4 compliance.
pid	PID	Process ID of the process. Required for POSIX and XPG4 compliance.
pgid	PGID	Process-group ID of the process. Required for POSIX and XPG4 compliance.
ppid	PPID	Process ID of the parent process. Required for POSIX and XPG4 compliance.
pri	PRI	Priority of the process (decimal).
rgroup	RGROUP	Group name associated with the real group ID of the process. Required for POSIX and XPG4 compliance.
ruser	RUSER	Login name associated with the real user ID of the process. Required for POSIX and XPG4 compliance.
size	SZ	Size (in blocks) of the core image of the process (decimal).
sz	SZ	Same as size.
state	S	The state of the process.
s	S	Same as state.
sctime	SCTIME	Cumulative system call time for the process.
stime	STIME	Starting time of the process. Not supported. Required for XPG4 compliance.
swaps	SWAPS	Number of times the process has been swapped.
time	TIME	Same as cputime. Required for POSIX and XPG4 compliance.
tty	TT	Controlling terminal for the process (if any). Required for POSIX and XPG4 compliance.
tt	TT	Same as tty.
uid	UID	User ID number of the process.
user	USER	Login name associated with the effective user ID of the process. Required for POSIX and XPG4 compliance.
vsz	VSZ	Size (in kilobytes) of the core image of the process (decimal). Required for POSIX and XPG4 compliance.
wchan	WCHAN	The event for which the process is waiting or sleeping.
waitswap	WAITSWAP	Cumulative time that the process has been swapped.

NOTES

Only an appropriately authorized user can see output for processes whose active security label is greater than that of the user.

If this utility is installed with a privilege assignment list (PAL), a user who is assigned the following privilege text upon execution of this command is allowed to perform the action shown:

Privilege Text	Action
`showall`	Allowed to see output for all processes. Allowed to successfully use the `-n` option.

If the `PRIV_SU` configuration option is enabled, the super user is allowed to see output for all processes and is allowed to successfully use the `-n` option.

EXIT STATUS

If `ps` finds processes to report, the exit status is 0; otherwise, it is 1.

BUGS

Some data printed for defunct processes are irrelevant.

When you use the `-f`, `-l`, or `-o` options, output may be wider than the allowed field, which causes the columns not to line up.

FILES

`/unicos`	System namelist
`/dev/mem`	Memory
`/dev/mpp/config`	CRAY T3D configured as part of the system
`/etc/passwd`	Supplies UID information
`/etc/group`	Supplies GID information
`/etc/ps_data`	Internal data structure
`/dev`	Searched to find terminal (tty) names
`/usr/include/sys/proc.h`	Supplies flag information

SEE ALSO

`kill`(1), `nice`(1), `privtext`(1), `renice`(1), `tty`(1)

`mppstat`(8) in the *CRAY T3D Administrator's Guide*, Cray Research publication SG–2507

NAME

`ptyrecon` – Manages pty reconnection

SYNOPSIS

```
ptyrecon -c [pty]
ptyrecon -d [pty]
ptyrecon -e [-t secs] [pty]
ptyrecon -h pty
ptyrecon -s
ptyrecon -S
```

IMPLEMENTATION

Cray PVP systems

DESCRIPTION

The `ptyrecon` utility enables or disables pseudo tty (pty) disconnection, and reconnects, searches, or hangs up disconnected sessions.

If disconnection is enabled, a session does not disappear on a pty master side close operation (such as `telnet close`), but remains disconnected for a specified amount of time. Users can later search, reconnect to, or hang up disconnected sessions.

Because this utility restores terminal characteristics, it permits the use of either cooked (line) or raw (character) terminal modes, like `vi` to disconnect sessions.

The `ptyrecon` utility accepts the following options:

`-c` [*pty*] Connects to an already disconnected pty. If *pty* is omitted, the current controlling terminal is used.

`-d` [*pty*] Disables reconnection on a pty. If *pty* is omitted, the current controlling terminal is used.

`-e` [`-t` *secs*] [*pty*]
Enables reconnection on a pty. `-t` *secs* indicates the number of seconds the disconnected session is to stay alive. If this argument is omitted, the default `DISTIMEO` in the `sys/ptyrecon.h` file is used. If *pty* is omitted, the current controlling terminal is used.

`-h` *pty* Hangs up the specified pty that is already disconnected.

`-s` Searches and displays owner's disconnected ptys.

`-S` Searches and displays all disconnected ptys (authorized users only).

NOTES

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
`system`, `secadm`	Allowed to search and display all disconnected ptys.

If the `PRIV_SU` configuration option is enabled, the super user is allowed to search and display all disconnected ptys.

To indicate a pty, use a number (such as 2, 02, or 002).

EXAMPLES

```
birch28% telnet cool
   ...
cool% ptyrecon -e
cool% bc
3 + 2
5
^]
telnet> close
Connection closed.
birch28% telnet cool
   ...
cool% ptyrecon -s
    PTY    UID    PID     TIME     TLIMIT
    000  10208   1629   0:00:51   0:10:00
cool% ptyrecon -c 000
5 + 8
13
quit
cool% logout
Connection closed by foreign host.
birch28%
```

SEE ALSO

`ptyrecon`(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

NAME

pwd – Displays working directory name

SYNOPSIS

```
pwd
```

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The pwd utility writes the absolute path name of the current working directory to standard output.

NOTES

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
system, secadm	In a privileged administrator shell environment, allowed to write shell-redirected output to any file.
sysadm	Shell-redirected output is subject to security label restrictions.

If the PRIV_SU configuration option is enabled, the super user can write shell-redirected output to any file.

EXIT STATUS

The pwd utility exits with one of the following values:

0 Successful completion.

>0 An error occurred.

MESSAGES

`Cannot open ..`
or
`Read error in ..` This message indicates possible file system trouble; you should try the full path name (`cd /`*`name/name/`*), followed by issuing a pwd command. It also may indicate that the `..` directory has been removed or the mode of the `..` directory does not allow reading.

SEE ALSO

cd(1)

NAME

quota – Reports quota information

SYNOPSIS

quota [-A] [-B] [-D] [-E] [-G] [-U] [-b] [-n] [-a *account_name*] [-f *file_quota*] [-g *group_name*] [-i *inode_quota*] [-p *fstab_path*] [-q *quota_file_name*] [-r *level*] [-s *file_system*] [-u *user_name*] [-v]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The quota utility reports on the state of the quota control system for the user or system administrator. Ordinary users are able to see information associated with their user ID (uid) and all authorized account IDs (acids) and group IDs (gids). An appropriately authorized user can see all acids, gids, and uids. A number of special exit codes can be used to determine the state of the quota system in scripts or batch jobs. Output is sent to the stdout file.

The quota utility accepts the following options:

-A Selects all acids. If -E is not present, all acids found in the /etc/udb entry for the user are selected; otherwise, all acids are selected. This option is not valid with -a.

-B Backup format. This option causes the report containing quota, warning, and usage information to be written in quadmin(8) directive format. If this option is used with -E, an ASCII file that represents the entire quota file is written. This option is not allowed with -b or -n.

-D Prints debug output to the standard error file.

-E Super user only (but see -q). Selects every existing id. If -A, -G, or -U is not present, the result is the same as if options -AGU had been presented. If any of the options -A, -G, or -U is present, only those classes of ids are selected.

-G Selects all gids. If -E is not present, all gids found in the /etc/udb entry for the user are selected; otherwise, all gids are selected. This option is not valid with -g.

-U Selects all uids. If -E is not present, only the user's current uid is selected; otherwise, all uids in /etc/udb are selected. This option is not allowed with -u.

-b Back-up format. This option causes the report containing quota and warning information to be written in quadmin(8) directive format. If this option is used with -E, an ASCII file that represents the entire quota file is written. This option is not allowed with -B or -n. Unlike the -B option, which also specifies back-up format, this option does not write out the usage information.

`-n` This option suppresses the report and causes just the name or names of the selected file systems (`-s` option) or quota files (`-q` option) to be written to the `stdout` file. This option is not allowed with `-b` or `-B`.

`-a` *account_name*
Selects the specified account names or `id`s. *account_name* is a list of one or more account names (or `acid`s) separated by commas or white space. Lists separated by white space must be enclosed in quotation marks so that the shell does not break them up into separate options. If the name is not an account name and it is numeric, it is treated as an account ID. The default is the `acid` of the current process, returned from `acctid`(2). This option is not valid with `-A`.

`-f` *file_quota*
Selects quota entries with at least *file_quota* blocks free in the quota. The default for *file_quota* is 0. This option is not allowed with `-r`.

`-g` *group_name*
Selects the specified groups. *group_name* is a list of one or more group names (or `gid`s) separated by commas or white space. Lists separated by white space must be enclosed in quotation marks so that the shell does not break them up into separate options. If the name is not a group name and it is numeric, it is treated as a `gid`. The default is the `gid` returned from `getgid` (see `getuid`(2)). This option is not valid if `-G` is selected.

`-i` *inode_quota*
Selects quota entries with at least *inode_quota* inodes free in the quota. The default for *inode_quota* is 0. This option is not allowed with `-r`.

`-p` *fstab_path*
Specifies an alternate path for `fstab`. Without this option, the default path is `/etc`, but this option allows the user to have a local version of these files for testing or experimentation. This option is intended for test purposes and is not recommended for normal use.

`-q` *quota_file_name*
Selects the named quota file. This option is intended to be used only if quota control is not active on a file system and can be used to bypass the normal `quotactl`(2) access method. Usually, only the super user will have permission to read the quota control file in this manner. If quota control is actively using this file, the content of the report may be inaccurate because the kernel could have information in its tables that have not been written to the file. This option is not valid with the `-s` option unless the `-b` or the `-B` option is also used. Because the quota file permission controls access in this mode, the super-user-only restriction on the other options is not applied.

`-r` *level* Selects quota entries specified by *level*. *level* selects quota entries below warning (`b`), at or above warning but below limit (`w`), or at limit (`l`) usage levels. These flags may be combined. The default is `blw`, which means all quota entries otherwise selected. This option is not allowed with `-f` or `-i`.

`-s` *file_system*
: Selects the named file system. The default is all file systems to which the other selection criteria apply. This option is not valid with the `-q` option unless the `-b` or the `-B` option is also used. This option suppresses the read of the kernel mount table to determine the list of mounted file systems.

`-u` *user_name*
: Selects the specified user names. *user_name* is a list of one or more user names (or `uid`s) separated by commas or white space. Lists separated by white space must be enclosed in quotation marks so the shell does not break them up into separate options. If the name is not a user name and it is numeric, it is treated as a user ID. The default is the `uid` returned from the `getuid`(2) system call. This option is not valid with `-U`. Only the super user may specify a `uid` different from that returned from `getuid`(2).

`-v`
: Reports explicit records, even if they have no current usage.

NOTES

If this utility is installed with a privilege assignment list (PAL), a user who is assigned the following privilege text upon execution of this command is allowed to perform the action shown:

Privilege Text	Action
`showall`	Allowed to see all user, group, and account IDs.

If this utility is installed with a PAL, a user with one of the following active categories is allowed to perform the action shown:

Active Category	Action
`system`, `secadm`, `sysadm`	Allowed to see all user, group, and account IDs.

If the `PRIV_SU` configuration option is enabled, the super user is allowed to see all user, group, and account IDs.

EXIT STATUS

A set of exit codes is available for use in scripts or batch jobs. The codes marked with `-f`, `-i`, or `-r` options appear only if those options are specified.

Code	Meaning
`0`	Normal completion
`1`	Usage warning; see the `stderr` file
`2`	Usage error
`3`	Internal error
`4`	Access denied
`5`	Quota system not active
`6`	Quota file magic mismatch
`7`	Quota file size is wrong

17 Warning level (-r)
18 Limit level (-r)
19 Both warnings and limit (-r)
20 Insufficient quota (-f, -i)

Report Format

The format of the normal report (when -n is not selected) is shown in this subsection, followed by an example. The portion in the outer bracket is repeated for each file system found to contain information for a selected id; the portion in the inner bracket is repeated for each id on that file system. The flags for quota warning and limit appear only if those conditions exist. The percentage shown with Quota is (Usage / Quota), and the percentage shown with Warning is (Usage / Warning). Percentages greater than 999.9 are shown as ***.*. If the -q option had been specified, quota file would replace file system in the example shown. When a * appears after a File block, Inode quota, or Warning value, the value shown is the default and has not been explicitly set.

The lines beginning with ** appear only if the particular warning, block, or unblock condition applies. The messages dealing with block and unblock times appear only on file systems with soft quotas and are intended to tell the user that, if the present usage levels are maintained, further file space will be denied or permitted at the time shown. These times are a prediction and should be considered approximate, especially if the file space in use changes appreciably.

```
⌈   File system: file_system_name
|
| ⌈ Account: account_name, Group: group_name, User: user_name, Id: 60000
| | ** Account warning time: Mon Feb 12 14:11 CST 1990
| | ** Account quota will [un]block on Mon Feb 12 14:11 CST 1990
| | ** Group warning time: Mon Feb 12 14:11 CST 1990
| | ** Group quota will [un]block on Mon Feb 12 14:11 CST 1990
| | ** User warning time: Mon Feb 12 14:11 CST 1990
| | ** User quota will [un]block on Mon Feb 12 14:11 CST 1990
| |                File blocks (512 bytes)   Inodes
| | Account Quota: 999999999999999* (100.0%) 9999999999* (100.0%) ** LIMIT
| |       Warning: 999999999999999* (100.0%) 9999999999* (100.0%) ** WARNING
| |         Usage: 999999999999999           9999999999
| |   Group Quota: 999999999999999* (100.0%) 9999999999* (100.0%) ** LIMIT
| |       Warning: 999999999999999* (100.0%) 9999999999* (100.0%) ** WARNING
| |         Usage: 999999999999999           9999999999
| |    User Quota: 999999999999999* (100.0%) 9999999999* (100.0%) ** LIMIT
| |       Warning: 999999999999999* (100.0%) 9999999999* (100.0%) ** WARNING
| |         Usage: 999999999999999           9999999999
```

The following is an example of a report showing the state of the quota control system for two users. No warning, block, or unblock conditions apply. Online quotas are in effect.

```
File system: /.fs/f04
 User: user1, Id: 001
                     File blocks (512 bytes)   Inodes
     User Quota:         4000000  (  4.0%)      50000  ( 69.0%)
        Warning:         3600000* (  4.4%)      45000* ( 76.6%)
          Usage:          158952                34477

 User: user2, Id: 002
                     File blocks (512 bytes)   Inodes
     User Quota:         4000000  (  6.9%)      50000  ( 69.0%)
        Warning:         3600000* (  7.6%)      45000* ( 76.6%)
          Usage:          274056                59316

File system: /.fs/f12
 User: user1, Id: 001
                     File blocks (512 bytes)   Inodes
     User Quota:         1300000* (  0.0%)      50000* (  0.0%)
        Warning:         1170000* (  0.0%)      45000* (  0.0%)
          Usage:               1                    1
```

The column labeled File blocks shows the quota block counts and block usage for online quotas. If "aggregate" quotas are in effect on the file system, the column will be labeled Aggregate blocks. In that case, the usage refers to the sum of the data blocks online and the data blocks migrated offline by the Data Migration Facility (DMF).

In the (default) online quotas case, files migrated offline by DMF do not count against the quota block counts. If the administrator has selected aggregate quotas, both online files and offline files contribute to the quota block count.

FILES

`/etc/acid`	Account file containing account ID and account name for each account
`/etc/group`	Group file containing group names and group IDs
`/etc/udb`	User validation file containing user control limits
`/etc/udb.public`	Public version of the `/etc/udb` file
`*/.Quota60`	Quota control file
`/etc/fstab`	File describing file system and swapping partitions used by UNICOS

SEE ALSO

`acctid`(2), `getuid`(2), `quotactl`(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

`quadmin`(8), `qudu`(8) in the *UNICOS Administrator Commands Reference Manual*, Cray Research publication SR–2022

UNICOS Resource Administration, Cray Research publication SG–2302

NAME

quotamon – Monitors UNICOS quota state

SYNOPSIS

quotamon [-s *timeout*] [-p]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The quotamon utility runs as a background process to monitor the state of the quota control features of the UNICOS operating system and report to the user when quota warning and/or limit conditions occur. The messages are written to the stderr file.

The system administrator may elect to start this utility automatically for all users through the /etc/cshrc and /etc/profile files. If running this utility is the user's choice, do not alter /etc/cshrc and /etc/profile, but instead make known to the users that quotamon can be started by placing the utility in the user's home directory .login or .profile file.

The quotamon utility accepts the following options:

-s *timeout* The *timeout* is the amount of time in seconds that should elapse between successive messages about the same event. The default is 30 seconds, but any positive integer from 0 to the maximum allowed integer value is accepted.

-p When the -p option is specified, the process ID of the running quotamon process is printed on standard output.

NOTES

You do not need to start the quotamon process. This feature is automatically provided by the Korn, standard, and C login shells.

BUGS

If a standard error message is issued from a user process and quotamon happens to write a message at the same time because of a quota signal, the messages could be intermixed in the stderr file.

Be careful not to initiate more than one copy of quotamon per job or interactive session. Additional copies wastefully occupy process table and memory space and cause duplicate messages to appear in the stderr file.

EXAMPLES

To start the quota monitor so that messages are not repeated within 60 seconds, enter the following line in the selected file or files mentioned previously. `quotamon` places itself in the background.

```
quotamon -s 60
```

SEE ALSO

`quota`(1)

NAME

rcp – Copies remote files

SYNOPSIS

rcp [-b] [-c *copy_buffer_size*] [-p] [-r] [-s *socket_buffer_size*] [-S *tos*] [-T] *sourcelist destination*

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The rcp utility copies files between machines. The *sourcelist* argument can refer to remote files, local files, or directories; arguments can consist of either absolute or relative path names.

Remote files are specified in the format *rhost*:*file* where *rhost* is a remote host name or alias (described in hosts(5)). When the login name differs from your login name on a Cray Research system, file names on a remote host are specified as *user*@*rhost*:*file*. If you do not specify a full path name, the path is interpreted relative to your login directory on *rhost*. A path name on a remote host can be quoted (using \, ", or ‘) so that metacharacters are interpreted remotely.

The local file name *file* may not contain a colon (:).

The rcp utility accepts the following options:

-b Displays the buffer sizes for the copy buffer and socket buffer during a transfer.

-c *copy_buffer_size*
Sets the copy buffer size. This buffer is used for reading and writing between the data socket and the source or destination file. rcp automatically chooses a copy buffer size; however, you can alter the selection. The -c option requires an argument. An argument of off turns off copy buffer sizing and the buffer defaults to the size specified by COPYBUFSIZE #define in the tcp_config.h file. An argument of auto sets buffer sizing to automatic and has no effect relative to default operation. A numeric argument sets the buffer to that size. The letter K or k can follow a numeric buffer size to specify a multiple of 1024. The default setting is auto. A size of 0 is synonymous with auto.

-p Preserves in its copies the modification times and access modes of the source files, ignoring the user file creation mode mask (see umask(1)). By default, the mode and owner of the copy of the file are preserved if it already existed; otherwise, the mode of the source file modified by the umask on the destination host is used.

-r Copies each subtree that is rooted at that name when any of the source files are directories. The destination must be a directory.

`-s` *socket_buffer_size*
: Sets the socket buffer size. This kernel buffer is for data transfer in the data socket. `rcp` automatically chooses a socket buffer size; however, you can alter the selection. The `-s` option requires an argument. An argument of `off` turns off socket buffer sizing and the buffer defaults to the default kernel socket buffer size. An argument of `auto` sets buffer sizing to automatic and has no effect relative to default operation. A numeric argument sets the buffer to that size. The letter `K` or `k` can follow a numeric buffer size to specify a multiple of 1024. The default setting is `auto`. A size of 0 is synonymous with `auto`.

`-S` *tos*
: Sets the IP Type-of-Service (TOS) option for the connection to the value *tos*, which can be a numeric TOS value or a symbolic TOS name that is found in the `/etc/iptos` file.

`-T`
: Outputs timing information after the transfer completes.

sourcelist
: Specifies one or more remote files, local files, or directories.

destination
: Specifies the destination file or directory.

The `rcp` utility does not prompt for passwords; your current local user name must exist on *rhost* and allow remote command execution by using `remsh`(1B). `rcp` requires that the `.rhosts` file exists in the `$HOME` directory of the remote host.

The system configuration may require the `/etc/hosts.equiv` and `.rhosts` files each to contain a match for the remote host, and also require the remote user and local user names to match.

The `rcp` utility handles third-party copies in which neither source nor target files are on the current machine. Host names can also take the form *rname*@*rhost* to use *rname* rather than the current user name on the remote host. The destination host name can also take the form *rhost*`.`*rname* to support destination machines that are running the 4.2BSD version of `rcp`.

NOTES

Use `rcp` with discretion in secure mode. For more information, see the *UNICOS Networking Facilities Administrator's Guide*, Cray Research publication SG–2304.

BUGS

When only a directory should be legal, the `rcp` utility does not detect all occurrences in which the target of a copy might be a file.

The `rcp` utility is confused by any output that is generated by commands in a `.profile`, `.login`, or `.cshrc` file on the remote host.

EXAMPLES

Example 1: To copy the `proposal` file, which is on a Cray Research system, into file `report` on the remote host `chemistry`, enter the following:

```
rcp proposal chemistry:report
```

Example 2: To copy the `whale` file, which is on the remote host `biology`, into file `mammal` on a Cray Research system, enter the following:

```
rcp biology:whale mammal
```

Example 3: The following example shows how to copy a file to or from an account that has a login other than your login on a Cray Research system.

In this example, the remote file `letter` must be accessed under the login name `tami`. To copy the file from the remote host `engineering` into a file named `memo` on a Cray Research system, enter the following:

```
rcp tami@engineering:letter memo
```

FILES

`$HOME/.rhosts`	File that contains a list of valid users
`/etc/hosts`	TCP/IP host name database

SEE ALSO

`ftp`(1B), `remsh`(1B), `rlogin`(1B), `umask`(1)

`hosts`(5), `rhosts`(5) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

Software Overview for Users, Cray Research publication SG–2052

TCP/IP Network User's Guide, Cray Research publication SG–2009

UNICOS Networking Facilities Administrator's Guide, Cray Research publication SG–2304

NAME

rdist – Maintains identical copies of files over multiple hosts

SYNOPSIS

/usr/ucb/rdist [-n] [-q] [-R] [-h] [-i] [-v] [-w] [-y] [-b] [-D] [-p] [-r] [-s] [-S *tos*] [-f *distfile*] [-d *var=value*] [-m *host*] [*names*]

/usr/ucb/rdist [-n] [-q] [-R] [-h] [-i] [-v] [-w] [-y] [-b] [-D] [-p] [-r] [-s] [-S *tos*] -c *names host*[.*login*][:*dest*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The rdist utility maintains identical copies of files over multiple hosts. It preserves the owner, group, mode, and last modification time of files, when possible, and can update programs that are executing.

rdist reads commands from *distfile* to direct the updating of files and directories.

The rdist utility accepts the following options and arguments:

-n Prints the commands without executing them. This option is used for debugging *distfile*.

-q Specifies quiet mode. The names of files that are being updated usually are printed on standard output. The -q option suppresses this.

-R Removes extraneous files. If a directory is being updated, any files that exist on the remote host but not in the master (local) directory are removed. This option is useful for maintaining truly identical copies of directories.

-h Follows symbolic links. Copies the file to which the link points rather than copying the link itself.

-i Ignores unresolved links. rdist usually tries to maintain the link structure of files being transferred and warns the user if all links cannot be found.

-v Verifies that the files are up-to-date on all hosts. Any files that are out-of-date are displayed, but no files are changed and no mail is sent.

-w Specifies whole mode. The whole file name is appended to the destination directory name. Usually, only the last component of a name is used when renaming files. This preserves the directory structure of the files being copied, rather than flattening the directory structure (for example, renaming a list of files such as (dir1/f1 dir2/f2) to dir3 would create files dir3/dir1/f1 and dir3/dir2/f2 rather than dir3/f1 and dir3/f2).

`-y` Specifies younger mode. If their *mtime* and *size* (see `stat`(2)) disagree, files usually are updated. The `-y` option prevents `rdist` from updating files that are newer than the master copy. Use this mode to prevent newer copies on other hosts from being replaced. A warning message is printed for files that are newer than the master copy.

`-b` Specifies a binary comparison. Performs a binary comparison and updates files if they differ, rather than comparing dates and sizes.

`-D` Invokes debug mode. `rdist` generates debugging information. Use this mode only if you must debug `rdist`.

`-p` Invokes force directory permissions mode. This option forces an update of all remote directories to have the same permissions as the master directory.

`-r` Follows symbolic links on the remote host. If the master is not a symbolic link, a symbolic link usually is overwritten.

`-s` Specifies short name mode. Compares only the first 14 characters of the trailing component of the file name when removing files that use `-R`.

`-S` *tos* Sets the IP Type-of-Service (TOS) option for the connection to the value *tos*, which can be a numeric TOS value or a symbolic TOS name found in the `/etc/iptos` file.

`-f` *distfile* Specifies the name of the file that contains the commands that direct the updating of the files and directories. If *distfile* is a dash (`-`), the standard input is used. If you omit this option, the program looks in the current directory first for `distfile` and then for `Distfile` to use as the input. If neither `distfile` nor `Distfile` is found, it uses the `-c` option.

`-d` *var=value*
Defines or overrides a variable (*var*) in the *distfile* with *value*. *value* can be an empty string, one name, or a list of names enclosed in parentheses and separated by tabs or spaces.

`-m` *host* Specifies machines to be updated. You can specify multiple `-m` options to limit updates to a subset of the hosts listed in the *distfile*.

names Specifies names of files to be updated or labels of commands to execute. If label and file names conflict, a label is assumed. If you omit this argument, `rdist` updates all the files and directories that are listed in *distfile*.

`-c` *names host*[.*login*][:*dest*]
Forces `rdist` to interpret the remaining arguments as a small *distfile*. The equivalent *distfile* is as follows:

(*names* ...) -> *host*[`.`*login*] `install` [*dest*] ;

Creating the *distfile*

The *distfile* contains a sequence of entries to do the updating that specifies the files to be copied, the destination hosts, and identifies the operations to perform. Each entry has one of the following formats:

<variable name> = *<namelist>*
[*label*:] *<source list>* -> *<destination list>* *<command list>*
[*label*:] *<source list>* :: *<time_stamp file>* *<command list>*

The first format is used for defining variables. The second format is used for distributing files to other hosts. The third format is used for distributing files that have changed since a given date.

The elements of the *distfile* are as follows:

<variable name> = <namelist>
Defines the files to be copied.

label Identifies a command for partial updates; labels are optional.

source list and *destination list*
The *source list* specifies a list of files and directories on the local host that will be used as the master copy for distribution. The *destination list* is the list of hosts to which these files will be copied. If the file is out-of-date on the host that is being updated (second format) or the file is newer than the time-stamp file (third format), each file in the source list is added to a list of changes. The source and destination lists have one of the following formats:

<name>
'(' *<zero or more names separated by white space>* ')'

command list The command list consists of zero or more of the following commands:

`install` *options opt_dest_name* ;
Copies out-of-date files and directories. Each source file is copied to each host in the destination list. Directories are recursively copied this way. *opt_dest_name* is an optional parameter to rename files. If an `install` command does not appear in the command list or you do not specify the destination name, the source file name is used. Directories in the path name are created if they do not exist on the remote host. To help prevent disasters, a nonempty directory on a target host is never replaced with a regular file or a symbolic link. However, under the `-R` option, a nonempty directory is removed if the corresponding file name is absent on the master host.

The *options* are `-R`, `-h`, `-i`, `-v`, `-w`, `-y`, `-b`, `-p`, `-r`, and `-s`; they have the same semantics as the options on the command line, except that they apply only to the files in the source list. The login name that is used on the destination host is the same as the local host, unless the destination name is of the format `host.login`.

`notify` *name list* ;
Mails the list of updated files (and any errors that occurred) to the listed names. If `@` does not appear in the name, the destination host is appended to the name (for example, `name1@host`, `name2@host`, ...).

`except` *name list* ;
Updates all of the files in the source list except the files listed in *name list*. This command usually is used to copy everything except certain files in a directory.

`except_pat` *pattern list* ;
Like the `except` command, except that *pattern list* lists regular expressions (see `ed`(1) for details). If one of the patterns matches a string within a file name, that file is ignored. Because \ is a quotation character, you must double it for it to become part of the regular expression. Variables are expanded in *pattern list*, but shell file pattern-matching characters are not expanded. To include a $, it must be escaped with a \.

`special` *name list string* ;
Specifies `sh`(1) commands that will be executed on the remote host after the file in *name list* is updated or installed. If you omit the *name list*, the shell commands are executed for each updated or installed file. The `FILE` shell variable is set to the current file name before executing the commands in *string*; *string* starts and ends with " and can cross multiple lines in *distfile*. You should separate multiple commands to the shell with a semicolon (;). Commands are executed in the user's home directory on the host being updated.

You cannot use the `special` command for rebuilding (for example, private databases) after a program is updated.

New-line characters, tabs, and blanks are used only as separators and are ignored otherwise. Comments begin with a # and end with a new-line character.

Variables to be expanded begin with a $ and are followed either by 1 character or a name enclosed in braces (see the EXAMPLES section).

The shell metacharacters are recognized and expanded (on the local host only) similar to `csh`(1). They can be escaped with a backslash. The ~ character also is expanded similar to `csh`, but it is expanded separately on the local and destination hosts. When you use the `-w` option with a file name that begins with ~, everything except the home directory is appended to the destination name. File names that do not begin with a `/` or ~ use the destination user's home directory as the root directory for the file name.

MESSAGES

A complaint of mismatching `rdist` version numbers can originate from starting your shell (for example, when you are in too many groups).

BUGS

Source files must reside on the local host in which `rdist` is executed.

You must execute a special command after all files in a directory are updated.

Variable expansion works only for name lists; a general macro facility is needed.

The `rdist` utility aborts files that have a negative *mtime* (before January 1, 1970).

A force option is needed to allow replacement of nonempty directories by regular files or symbolic links. A means of updating file modes and owners of otherwise identical files also is needed.

EXAMPLES

Following is an example of a *distfile*:

```
HOSTS = ( matisse arpa.root)

FILES = ( /bin /lib /usr/bin /usr/games
        /usr/include/{*.h,{stand,sys,vax*,pascal,machine}/*.h}
        /usr/lib /usr/man/man? /usr/ucb /usr/local/rdist )

EXLIB = ( Mail.rc aliases aliases.dir aliases.pag crontab dshrc
        sendmail.cf sendmail.fc sendmail.hf sendmail.st uucp vfont )
${FILES} -> ${HOSTS}
        install -R ;
        except /usr/lib/${EXLIB} ;
        except /usr/games/lib ;
        special /usr/lib/sendmail "/usr/lib/sendmail -bz" ;

srcs:
/usr/src/bin -> arpa
        except_pat ( \\.o\$ /SCCS\$ ) ;

IMAGEN = (ips dviimp catdvi)

imagen:
/usr/local/${IMAGEN} -> arpa
        install /usr/local/lib ;
        notify ralph ;

${FILES} :: stamp.cory
        notify root@cory ;
```

FILES

`distfile`	Input command file.
`/tmp/rdist*`	Temporary file for update lists.
`$HOME/.rhosts`	User file that lists remote host names and logins names of users who are allowed access to the home directory. This file must be present on the remote host.

SEE ALSO

csh(1), ed(1), sh(1)

stat(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

NAME

`read` – Reads a line from standard input

SYNOPSIS

`read` [`-p`] [`-r`] [`-s`] [`-u` *n*] *var...*

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4
AT&T extensions (`-p`, `-s`, and `-u` options)

DESCRIPTION

The `read` utility reads one line from standard input.

By default, unless the `-r` option is specified, backslash (\) acts as an escape character. If standard input is a terminal device and the invoking shell is interactive, `read` prompts for a continuation line when the following conditions are met:

- The shell reads an input line that ends with a backslash, unless the `-r` option is specified.
- A here document is not terminated after a `<newline>` is entered.

The `read` utility accepts the following options and operand:

`-p` Causes the input line to be taken from the input pipe of a process spawned by the shell by using `|&`.

`-r` Does not treat a backslash character in any special way (considers each backslash to be part of the input line).

`-s` Saves the input as a command in the shell history file.

`-u` *n* Reads from the one-digit file descriptor unit *n*.

var... Specifies the name of an existing or nonexisting shell variable.

The line is split into fields as in the shell; the first field is assigned to the first variable *var1*, the second field to the second variable *var2*, and so on. If fewer *var* operands are specified than there are fields, the leftover fields and their intervening separators are assigned to the last *var*. If fewer fields than *var*s exist, the remaining *var*s are set empty strings.

The setting of variables specified by the *var* operands affect the current shell execution environment.

NOTES

The `read` utility is a built-in utility to the standard shell (`sh`(1)). An executable version of this utility is available in `/usr/bin/read`.

EXIT STATUS

The `read` utility exits with one of the following values:

0 Successful completion.

>0 End-of-file was detected or an error occurred.

EXAMPLES

Example 1: The following command prints a file with the first field of each line moved to the end of the line:

```
$ while read -r xx yy
> do
>       printf "%s %s\n" "$yy" "$xx"
> done < input_file
```

SEE ALSO

`sh`(1), `line`(1)

NAME

regcmp – Compiles regular expressions

SYNOPSIS

regcmp [-] *files*

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

Usually, the regcmp utility precludes the need for calling regcmp(3C) from C programs. This saves on both execution time and program size. The regcmp utility compiles the regular expressions in *file* and places the output in *file*.i.

The regcmp utility accepts the following options:

- Output will be placed in *file*.c.

files Files to compile.

The format of entries in *file* is a name (C variable), followed by one or more blanks, followed by a regular expression enclosed in double quotation marks.

The output of regcmp is C source code. Compiled regular expressions are represented as char[] arrays. Thus, the *file*.i files may be included in C programs, or *file*.c files may be compiled and later loaded. In a C program that uses the regcmp output, regex (abc, line) applies the regular expression abc to line.

EXAMPLES

Example 1: The following are examples of the contents that may be found in the input file for regcmp:

```
name        "([A-Za-z][A-Za-z0-9_]*)$0"

telno       "({0,1}([2-9][01][1-9])$0){0,1} *"
            "([2-9][0-9]{2})$1[ -]{0,1}"
            "([0-9]{4})$2"
```

Example 2: In a C program that uses the regcmp output, the following will apply the regular expression named telno to line.

```
regex(telno, line, area, exch, rest)
```

SEE ALSO

regcmp(3C) in the *UNICOS System Libraries Reference Manual*, Cray Research publication SR–2080

NAME

remsh, rsh – Invokes a remote shell

SYNOPSIS

/usr/ucb/remsh *host* [-d] [-l *username*] [-n] [-S *tos*] [*command*]
/usr/ucb/rsh *host* [-d] [-l *username*] [-n] [-S *tos*] [*command*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The remsh and rsh utilities connect to *host* and execute *command*. The command copies its standard input to the remote command; copies the standard output of the remote command to its standard output; and copies the standard error of the remote command to its standard error. Interrupt, quit, and terminate signals are propagated to the remote command; usually the remote shell terminates when the remote command terminates. remsh requires that the .rhosts file exists in the $HOME directory of the remote host.

The remsh and rsh utilities accept the following options:

-d Sets the SO_DEBUG option on the sockets for the standard output and standard error.

-l *username* Specifies the account name (*username*) to use when loggin in to the remote machine. The default is the current account name. The remote name must be authorized for automatic authentication (as described in rlogin(1B)) to the originating account; no provision is made for specifying a password with a command.

-n Redirects the input of remsh or rsh to /dev/null if no input is desired.

-S *tos* Sets the IP Type-of-Service (TOS) option for the connection to the value *tos*, which can be a numeric TOS value or a symbolic TOS name found in the /etc/iptos file.

command Specifies the shell command to be executed. If you omit *command*, rather than executing a single command, you will be logged in on the remote host by using rlogin(1B).

Shell metacharacters that are not enclosed in quotation marks are interpreted on the local machine; metacharacters that are enclosed in quotation marks are interpreted on the remote machine. Thus, the following command line appends the remote file remotefile to the local file localfile:

```
remsh otherhost cat remotefile >> localfile
```

The following command line appends remotefile to remote file otherremotefile:

```
remsh otherhost cat remotefile ">>" otherremotefile
```

Host names are given in hosts(5). Each host has one standard name that is the first name specified in the file. This name is rather long and unambiguous; therefore, each host name may have one or more nicknames. The host names for local machines are also commands in the /usr/hosts directory; when you put this directory in your search path, you can omit rsh and remsh.

NOTES

The remsh utility is called rsh on UNIX 4.2 BSD systems.

The rsh(1) utility, the restricted shell, is supported in /bin. The /bin/rsh and /usr/ucb/rsh are two entirely different utilities.

If the remote system is using certain security features, the system configuration can require the /etc/hosts.equiv and .rhosts files each to contain a match for the originating host, and requires the remote user and local user names to match. If the system is configured this way, the -l option is not allowed.

FILES

$HOME/.rhosts	On remote machine
/etc/hosts	TCP/IP host name database

SEE ALSO

rlogin(1B)

setsockopt(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

hosts(5), rhosts(5) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

TCP/IP Network User's Guide, Cray Research publication SG–2009

Software Overview for Users, Cray Research publication SG–2052

UNICOS Networking Facilities Administrator's Guide, Cray Research publication SG–2304

NAME

renice – Sets system scheduling priorities of running processes

SYNOPSIS

```
renice [-n increment] [-g] ID ...
renice [-n increment] -j ID ...
renice [-n increment] -p ID ...
renice [-n increment] -u ID ...
```

Obsolescent version; may not be supported in future releases:

```
renice nice_value pid ... [-g gid ...] [-j jobid ...] [-p pid ...] [-u user ...]
renice nice_value [-g gid ...] [-j jobid ...] [-p pid ...] [-u user ...]
```

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4
CRI extensions (-j option)

DESCRIPTION

The renice utility requests that the system scheduling priorities of one or more running processes be changed. When a process ID is reniced (the default), the applicable processes are specified by their IDs. When a process group is reniced, the request applies to all processes in the process group.

If the requested *increment* (or *nice_value*) raises or lowers the system scheduling priority of the executed utility beyond the system limits, then the limit whose value was exceeded is used. The *increment* (or *nice_value*) refers to the system's nice value. The nice value can range from 0 through 39; 0 represents the highest priority, and 39 represents the lowest priority.

The renice utility the following options and operands:

-n *increment* Specifies how the system scheduling priority of the specified process(es) is adjusted.

-g Interprets all arguments (or just the *gid* arguments in the obsolescent version) as process group IDs.

-j Interprets all arguments (or just the *jobid* arguments in the obsolescent version) as job IDs. The *increment* argument is a positive or negative integer that modifies the system scheduling priority of the specified process(es). If no -n *increment* is specified, the default *increment* is 4.

-p Interprets all arguments (or just the *pid* arguments in the obsolescent version) as process IDs. If no options are specified, the -p option is the default.

`-u` Interprets all arguments (or just the *user* arguments in the obsolescent version) as users. If a user exists with a user name equal to the operand, the user ID of that user is used; otherwise, the operand represents an unsigned integer and interprets it as the numeric user ID of the user.

ID Specifies a process ID, a job ID, a process group ID, or user name or user ID, depending on the option selected.

In the obsolescent version, the `-g`, `-j`, `-p`, and `-u` options can each take multiple arguments. A *pid* of 0 means the current process, a *jobid* of 0 means the current job, a *gid* of 0 means the current process group, and a *user* of 0 means the current user.

nice_value (Obsolescent) Saves the value specified as the actual system scheduling priority, rather than as an increment to the scheduling priority. Specifying a scheduling priority higher than that of the existing process may require super-user privileges.

NOTES

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
`system`, `secadm`	Allowed to set the scheduling priority of any process. In a privileged administrator shell environment, shell-redirected I/O is not subject to file protections.
`sysadm`	Allowed to set the scheduling priority of any process subject to security label restrictions. Shell-redirected I/O is subject to security label restrictions.

If the `PRIV_SU` configuration option is enabled, the super user is allowed to set the scheduling priority of any process. Shell-redirected I/O on behalf of the super user is not subject to file protections.

EXIT STATUS

The `renice` utility exits with one of the following values:

0 All processes had their priorities successfully changed.

>0 An error occurred.

SEE ALSO

`nice`(1), `ps`(1)

`nice`(2), `nicem`(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

NAME

resize – Sets terminal settings current window size

SYNOPSIS

resize [-c] [-u] [-s [*row col*]]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The resize utility writes to its standard output commands for setting the TERM and TERMCAP environment variables to indicate the current size of a window from which the command is run.

The resize utility accepts the following options:

-c	Causes the commands to be formed appropriately for csh(1) rather than sh(1).
-u	Causes the commands to be formed appropriately for sh (the standard shell) rather than csh.
-s [*row col*]	Uses Sun tty escape sequences. A new *row* and *column* size may be specified; the window will resize appropriately. If -s is not specified, VT102 escape sequences are used.

BUGS

The -u must appear to the left of -s, if both are specified.

The -c must appear to the left of -s, if both are specified.

There should be some global notion of display size; termcap and terminfo need to be analyzed in the context of window systems.

EXAMPLES

The following aliases, when executed as a command, reset the environment of the current shell:

```
alias xs       ´eval `resize`´
alias xrs       ´set noglob; eval `resize -s \!\*`´
```

The following example sets your TERM to sun, exports TERM, and then inserts the size of your sun screen into the environment:

```
TERM=sun;export TERM;eval ‘resize -s‘
```

FILES

`~/.shrc`	User's alias for the command

SEE ALSO

`csh`(1), `sh`(1), `tset`(1B)

NAME

`restart` – Recovers a process, multitask group, or job from a restart file

SYNOPSIS

`restart` [`-i`] [`-d`] [`-f`] [`-w`] *file*

IMPLEMENTATION

Cray PVP systems

DESCRIPTION

The `restart` utility recovers the process, multitask group, or job defined in the named restart file.

The `restart` utility accepts the following options:

`-i` Causes `restart` to print the ID returned by the `restart`(2) system call on standard output.

`-d` Gives the restarted processes the Distributed Computing Environment (DCE) credentials of the current process rather than using the ones stored in the restart file.

`-f` Forces recovery, even if one or more of the files referenced by the restart image has changed.

`-w` Causes `restart` to wait until the restarted process or job completes.

file Specifies the restart file. The *file* operand is required.

NOTES

The following restrictions apply to processes and jobs that are to be restarted:

- Only an appropriately authorized user may checkpoint or restart another user's job.
- Processes using online tapes cannot be checkpointed or restarted.
- If a user attempts to copy a restart file, it is no longer marked as a restart file and is not restartable.
- If a user attempts to move a restart file to a different file system, it is no longer marked as a restart file and is not restartable.
- If an interactive session is checkpointed and later recovered with a `restart` utility, all processes that are part of the session which performed the restart are terminated.

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
`system`, `secadm`	Allowed to restart any file.
`sysadm`	Allowed to restart any file, subject to security label restrictions.

If the PRIV_SU configuration option is enabled, the super user is allowed to restart any file.

EXAMPLES

The following example illustrates how to use the restart utility to recover a process that has been checkpointed.

The user is running a.out in the background and enters a chkpnt(1) utility to checkpoint the process. The -p option specifies the PID and the -k option kills the process (pid.23634). The user then issues an ls(1) command to list information about pid.23634; the capital R at the beginning of the long listing indicates a restart file has been created.

```
unicos$ a.out &
23634

unicos$ chkpnt  -p  23634  -k

unicos$ ls  -l  pid.23634
Rr--------  1  (id) (group) (size) (date) pid.23634
```

Later, the user enters a restart utility to recover the process. The ps(1) listing confirms the process has been reactivated.

```
unicos$ restart  pid.23634

unicos$ ps
   PID     TTY  TIME COMMAND
  23634   p031  0:13 a.out
  23510   p031  0:00 sh
  23758   p031  0:00 ps
```

SEE ALSO

chkpnt(1), chkpnt_util(1), ps(1)

chkpnt(2), restart(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

General UNICOS System Administration, Cray Research publication SG–2301

NAME

rlogin – Invokes the remote login

SYNOPSIS

/usr/ucb/rlogin *rhost* [-d] [-e*c*] [-l *username*] [-8] [-E] [-L] [-S *tos*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The rlogin utility connects your terminal on the current local host system to the remote host system *rhost*. Your remote terminal type is the same as your local terminal type (as specified in your TERM environment variable). All echoing occurs at the remote site; therefore, the rlogin is transparent (except for delays). Flow control through <CONTROL-s> and <CONTROL-q> and flushing interrupt input and output are handled properly. A line of the form ~. disconnects from the remote host; ~ is the escape character. A line of the form ~<CONTROL-z> suspends the rlogin process, and a line of the form ~<CONTROL-y> suspends the send portion of the rlogin process, but allows output from the remote system. A line of the form ~z is the same as ~<CONTROL-z>.

The rlogin utility accepts the following options:

rhost	Specifies the remote host.
-d	Uses setsockopt(2) to turn on socket debugging on the TCP sockets that are used for communication with the remote host.
-e*c*	Specifies an escape character (*c*). No space can separate this option flag (-e) and the new escape character (*c*).
-l *username*	Specifies the account name (*username*) to use when logging in to the remote machine. Default is the current account name.
-8	Allows the transmission of 8-bit data.
-E	Stops any character from being recognized as an escape character. When used with the -8 option, this provides a completely transparent connection.
-L	Allows the rlogin process to be run in -opost mode (see stty(1)).
-S *tos*	Sets the IP Type-of-Service (TOS) option for the connection to the value *tos*, which can be a numeric TOS value or a symbolic TOS name found in the /etc/iptos file.

NOTES

The system configuration can require the `/etc/hosts.equiv` and `.rhosts` files each to contain a match for the originating host, and also require the remote user and local user names to match.

The `rlogin` requests are validated to ensure that the remote host or workstation security levels and compartments, as defined in the network access list (NAL), are within the security level range and are authorized compartments for the UNICOS system. The user security values are set to the most restrictive boundary conditions defined by the NAL and the user database (UDB).

The `rlogin` process also validates the user's right to access the UNICOS system from the host. Access to the UNICOS system is granted or denied based on the workstation access list (WAL) check the login process performs.

The results of user validation are recorded in the security log.

BUGS

Not enough terminal characteristics are propagated.

FILES

`$HOME/.rhosts`	Remote machine
`/etc/hosts`	TCP/IP host name database

SEE ALSO

`remsh`(1B), `stty`(1), `telnet`(1B)

`setsockopt`(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

`hosts`(5), `rhosts`(5) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

TCP/IP Network User's Guide, Cray Research publication SG–2009

Software Overview for Users, Cray Research publication SG–2052

UNICOS Networking Facilities Administrator's Guide, Cray Research publication SG–2304

NAME

rls – Releases reserved tape resources

SYNOPSIS

```
rls -a [-f] [-n] [-s]
rls -d dvn|all [-f] [-k] [-n] [-s]
rls -p pathname [-f] [-k] [-n] [-s]
```

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The rls utility releases resources reserved with the rsv(1) utility and breaks the association between the device and the *pathname* (see the -p and -P options of the tpmnt(1) utility).

You must specify -a, -d, or -p.

The rls utility accepts the following options:

-a — Releases all resources. -a is not valid with -d, -k, or -p. See the CAUTIONS section.

-f — Specifies that any scratch volumes from the autoloader scratch pool used during this tape session will be returned to the scratch pool for reuse.

-n — Specifies no-unload status for the tape. This option has the same effect as the -u option on the tpmnt(1) utility. This option is useful when a tape is used repeatedly.

-s — Specifies that any scratch volumes from the autoloader scratch pool used during this tape session will not be returned to the scratch pool. The user will keep these volumes.

-d *dvn*|all — Releases specified devices. For *dvn*, specify all to release all the used devices or specify a list of device names separated by a colon (:) so that only devices identified are released (see the tpstat(1) utility for information on how to find the device name). You cannot use the -d option with the -p option.

-k — Keeps resource privilege. If you specify the -k option, the process limit and current reservation are not decremented. You must specify a path name or a device to be released. You may issue an additional tpmnt(1) command without issuing an rsv(1) command.

-p *pathname* — Specifies a path name previously identified in a tpmnt(1) utility that has a -p or -P option. The request causes the tape equipment, which is associated with the path, along with the privilege to use the equipment, to be released back to the resource pool. You cannot use the -p option with the -d option.

CAUTIONS

The `rls -a` utility line releases all tapes reserved by your process ID and reserved tape resources, including reserved tape resources used by the processes running in the background. The `rls` utility releases resources that are closed; all other resources are put into a `releaseending` state until a `close`(2) system call has been executed for those resources. To release reserved tape resources for a specific process, use the `-p` or `-d` option.

EXIT STATUS

If `rls` completes successfully, 0 is returned; otherwise, a nonzero value is returned. Where possible, this exit status code is normalized to the last three digits. Exit status values are documented in the *Tape Subsystem User's Guide*, Cray Research publication SG–2051.

EXAMPLES

The following examples illustrate different uses of the `rls` utility:

Example 1: The `rls` utility releases devices `tape03` and `tape01`:

```
rls -d tape03:tape01
```

Example 2: The `rls` utility releases all of the devices that have been used:

```
rls -d all
```

Example 3: The `rls` utility releases the tape device that has the path name `tapfile`, which was specified with the `-p` option on `tpmnt`(1):

```
rls -p tapfile
```

Example 4: The `rls` utility releases the tape device that has the path name `tapfile`, allowing you to keep the tape resource and to perform another `tpmnt`(1) following it:

```
rls -p tapfile -k
```

Example 5: The `rls` utility releases all tape resources, does not unload any mounted volumes, and specifies that the autoloader scratch volumes will be saved for the user:

```
rls -a -n -s
```

SEE ALSO

`rsv`(1), `tpmnt`(1), `tprst`(1), `tpstat`(1)

`close`(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

Tape Subsystem User's Guide, Cray Research publication SG–2051

NAME

rm, rmdir – Removes files or directories

SYNOPSIS

```
rm [-f] [-i] [-r] [-R] files
rmdir [-p] [-s] dirnames
```

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4
AT&T extension (`rmdir -s` option)

DESCRIPTION

The `rm` utility removes the entries for one or more files from a directory. If an entry was the last link to the file, the file is destroyed. To remove a file, you must have write permission in its directory, but do not need read or write permission on the file itself. (See the note about MAC-protected files in the **NOTES** section.)

If a file has no write permission and the standard input is a terminal, the name and the full set of permissions (in octal) for the file are printed, followed by a question mark. This is a prompt for confirmation. If the answer begins with a lowercase `y` (for yes), the file is deleted; otherwise, the file remains.

If the standard input is not a terminal, `rm` will operate as if the `-f` option were in effect.

The `rmdir` utility removes the specified directories, which must be empty.

The `rm` utility accepts the following options:

`-f` Removes all specified files (ignores write-protection) in a directory without prompting the user. In a write-protected directory, or when MAC prevents writing the file, files are not removed, regardless of their permissions, and an error message is issued.

If a designated file is a directory, and the `-r` and `-R` options were not specified, an error message is issued, and `rm` will continue with any remaining file operands.

Any previous occurrences of the `-i` option are ignored.

`-i` Confirms removal of all files occurs interactively. To remove a file, your response must begin with a lowercase `y`. This option overrides the `-f` option and remains in effect even if the standard input is not a terminal.

Any previous occurrences of the `-f` option are ignored.

`-r`
`-R` Recursively removes any directories and subdirectories in the argument list. The directory is emptied of files and removed. Usually, the user is prompted for removal of any write-protected files in the directory. The write-protected files are removed without prompting if the `-f` option is used, or if the standard input is not a terminal and the `-i` option is not used. (See the note about MAC protected files in the NOTES section.)

files Files to be removed.

If you try to remove a nonempty, write-protected directory, `rm` will fail (even if you use the `-f` option), causing an error message.

If a directory to be removed is the current directory, `rm` silently ignores the current directory name and does not remove it.

The `rmdir` utility accepts the following options:

`-p` Lets you remove directory path *dirname* if it is empty, and its parent directories if they become empty. A message is printed on standard output as to whether the whole path is removed or part of the path remains for some reason.

`-s` Suppresses the message printed on standard error when `-p` is in effect.

dirnames Specifies the names of the directories to be removed.

NOTES

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
`system`, `secadm`	Allowed to remove any file.
`sysadm`	Allowed to remove any file, subject to security label restrictions on the file's path. Shell-redirected I/O is subject to security label restrictions.

If the `PRIV_SU` configuration option is enabled, the super user is allowed to remove any file.

When multi-level security is activated, mandatory access control (MAC) rules can prevent removing a file if MAC would prevent writing the file or the directory that contains it.

EXIT STATUS

The `rm` utility exits with one of the following values:

0 If the `-f` option was not specified, all the named directory entries were removed successfully. If the `-f` option was specified, all the existing named directory entries were removed successfully.

>0 An error occurred.

The `rmdir` utility exits with one of the following values:

0 Each directory entry specified was removed successful.

>0 An error occurred.

SEE ALSO

`mkdir`(1)

`rmdir`(2), `unlink`(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

NAME

rmdel – Removes a delta from an SCCS file

SYNOPSIS

rmdel -r *SID files*

IMPLEMENTATION

All Cray Research systems

STANDARDS

XPG4

DESCRIPTION

The rmdel utility removes the delta specified by the source identifier (SID) from each named Source Code Control System (SCCS) file. The delta to be removed must be the newest (most recent) delta in its branch in the delta chain of each named SCCS file. In addition, the SID-specified delta must not be that of a version being edited for the purpose of making a delta (that is, if a *p-file* (see get(1) exists for the named SCCS file, the SID-specified delta must *not* appear in any entry of the *p-file*).

The rmdel utility accepts the following option and operands:

-r *SID* Specifies the SID (SCCS IDentification) level of the delta to be removed.

files Specifies the SCCS file from which the specified delta is removed. If a directory is named, rmdel behaves as though each file in the directory were specified as a named file, except that non-SCCS files (last component of the path name does not begin with s.) and unreadable files are silently ignored. If a name of - is given, the standard input is read; each line of the standard input is taken to be the name of an SCCS file to be processed; non-SCCS files and unreadable files are silently ignored.

If you make a delta, you can remove it; if you own the file and directory, you can remove a delta.

EXIT STATUS

The rmdel utility exits with one of the following exit values:

0 Successful completion.

>0 An error occurred.

MESSAGES

Use help(1) for explanations.

EXAMPLES

The following example removes delta `1.2` from the file `s.example.c`:

```
rmdel -r1.2 s.example.c
```

FILES

`x.`*file*	See `delta`(1)
`z.`*file*	See `delta`(1)

SEE ALSO

`admin`(1), `cdc`(1), `comb`(1), `delta`(1), `get`(1), `help`(1), `prs`(1), `sact`(1), `sccsdiff`(1), `unget`(1), `val`(1), `vc`(1), `what`(1)

`sccsfile`(5) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

NAME

rmgr – Provides an interface to the Unified Resource Manager (URM) daemon

SYNOPSIS

rmgr [-c *directive*] [-D] [-h *hostname*] [-I *directory*] [-l *username*] [-s *socket*]

rmgr [-D] [-h *hostname*] [-I *directory*] [-l *username*] [-s *socket*] [*file*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The rmgr utility accepts the following options:

-c *directive* Invokes any rmgr *directive*. When the -c option is used, the *file* argument is not allowed and the rmgr utility terminates after processing the -c *directive*.

-D Debug mode; used for testing purposes only. Writes internal information to the stderr file.

-h *hostname* Names the host on which the target URM is running. The default host is the local host. An environment variable, RMGR_HOSTNAME, may name the target host.

-I *directory* Specifies an alternate directory in which to look for include files. If the file named on an include directive is not found using its name as written, rmgr checks for the file in the *directory* named with -I. Only one -I option can be used.

-l *username* Specifies a different user name by which to connect to URM. Using this option will result in an anonymous connection to URM (nonprivileged). For the connection to be accepted, the specified *username* must be contained under the URM /admin/anonymous/ node.

-s *socket* Specifies a socket (or service found in /etc/services) name or port number on which the target URM is listening. The default service name is urm. An environment variable, RMGR_SOCKET, may name the socket or port number.

file Names a command file. If a name is not given, rmgr will read from the stdin file. If stdin is connected to an interactive device, a prompt will be issued so the user will know that input is needed. If the -c option is used, you cannot specify a *file*.

The rmgr interface includes the following subcommands, some of which may be used only by an authorized administrator. Subcommands can be specified with either an initial uppercase letter or an initial lowercase letter (for example, both Quit and quit are valid). Only the lowercase form is shown here.
NOTE: Additional subcommands exist but are undocumented; these undocumented subcommands are reserved for use by Cray Research.

`delete` *object* (Authorized administrator only) Removes objects from the object tree. To delete an object, you must have write permission to that object. The directive to delete object `/val/myval` is as follows:

```
delete /val/myval
```

The following restrictions apply to what can be deleted:

The root node cannot be deleted.
No object with an existing synonym can be deleted.
No object with a lock can be deleted.
No resource object assigned to any job can be deleted.
A node with child objects cannot be deleted.
Only objects you own, or to which you have write permission, can be deleted.

`include` *file* Switches input file to *file*. After the named file has been read to the end-of-file, reading reverts to the line following this line. Includes can be nested up to 30 levels deep. To include a file named `setup/times`, use the following directive (the file name must appear in quotes):

```
include "setup/times"
```

`quit` Disconnects `rmgr` from the URM daemon and terminates the `rmgr` session.

`repeat` As a prefix to a directive, `repeat` causes `rmgr` to reissue a URM directive every 10 seconds and display the result. For example, to see the machine load changing:

```
repeat view /machine/load
```

`set` *parameter* Changes URM configuration parameter. Parameters include the following:

Parameter	Description
`basenode`	Changes the start of a name search from the root of the object tree to some other starting node (similar to the UNICOS `cd`(1) command). When this directive is set, any object name not beginning with a slash character (`/`) will be found starting at the named node, rather than the root node. An example of this directive follows: `set basenode` *object_name* The named object must be of type `Node` and must exist in the object tree. If the named object is removed while it is the base node, the base node is changed to root.

`defperm` (Authorized administrator only) Sets the default permission for newly created objects. When a user is connected to URM, the default permission is set to `rwr-`. This directive allows that default to be altered. Permissions are read from left to right as "owner read, owner write, other read, other write." All four permission fields must always be stated. Examples of this directive include the following :

```
set defperm rwrw
set defperm r---
```

The first directive sets the default permission for new objects to read and write for every user. This means that anyone can change or delete this object. The second directive makes new objects have owner read only permission. The owner is allowed to remove an object regardless of its permission and the URM administrator can do anything with any object, regardless of its permission.

`lock` (Authorized administrator only) Sets the delete lock on an object. A locked object may not be removed and there is no provision to remove a lock. The lock is intended for essential objects having to do with machine loading information which, for performance reasons, have pointer references from various internal places in URM. An example of this directive follows:

```
set lock /machine/target/memory
```

If any child object of a node is locked, no node higher in the object tree (closer to the root) can be removed.

`log` (Authorized administrator only) Enables logging of directives and informative and error messages. Logging is enabled by default. A log message is always written when this directive occurs:

```
set log
```

`log` *directory* (Authorized administrator only) Enables logging and switches the log file to reside in *directory*. If the path names are different, the current log is terminated with a message and closed, and a new log file is opened in *directory*. Log file names cannot be changed. Error messages appear if *directory* is the same as the current path or if the new path is inaccessible or full. If a log file of today's name already exists in *directory*, new messages will be appended to it.

`nolog` (Authorized administrator only) Disables logging. A log message is written only when this directive changes the state from logging to no logging:

```
set nolog
```

`perm` (Authorized administrator only) Changes the permission of an existing object. Only the URM administrator or the owner of an object can change the permission of that object. For example, for an existing object named `/val/xyz`, its permission can be changed using the following directive:

```
set perm rw-- /val/xyz
```

No matter what its prior permission, the object now has `rw--` permission.

`stopdaemon` (Authorized administrator only) Sends a directive to URM to terminate, then `rmgr` executes `quit`.

`view` [*name*|*type*] *option*

Displays options. The optional *name* or *type* argument can be used to display the name or type, respectively, of an object. The following options are valid:

Option	Description
`eshare` *nnn* [*lnode_ID*]	Shows effective share for user name *nnn*
`help`	Shows a list of all `view` options. NOTE: The `view help` command displays additional `view` options that are undocumented; these undocumented options are reserved for use by Cray Research.
`jlist`	Shows the Joblist table
`jobs` *nnn*	Shows jobs for user name *nnn*
`jobs` *uuu*	Shows jobs for user ID (UID) *uuu*
`jpath`	Shows the Jobpath table
`jselect`	Shows the job selector
`restart` *nnn*	Shows `chkpnt` images for user name *nnn*
`restart` *nnn n*	Shows detailed content of `chkpnt` image number *n*
`restart` *uuu*	Shows `chkpnt` images for user ID *uuu*
`rpath`	Shows the Respath table
`sds`	Shows SDS management values
`tpath`	Shows the Timepath table
`users`	Shows the users chain
`/`*xxx*	Shows object or node chain `/`*xxx* (for example, `view /hosts`)

*/xxx/**	Shows objects below */xxx* (for example, `view /urm`)
xxx	Shows node relative to the current node

object = *exp* Value-type objects may be given a value when they are declared, but can also have their value changed, using the value assignment directive. Value-type objects include numeric values (`Float` or `Int`) and string values (`Str`). Time-type objects may not be given a value in this way.

If an integer object named `/val/myval` existed, to set its value to 1234, use the following directive:

```
/val/myval = 1234
```

If the object is type `Str`, only quoted strings and other objects of type `Str` can appear in the expression. You can assign a value to the object `/val/mystring` as follows:

```
/val/mystring = "A string"
```

The quotation marks will not appear in the strings.

To change the minimum rank for a batch job, use the `usetjob`(8) command.

Checkpointing

URM can be configured to checkpoint individual sessions if requested by the owner of the session. The session owner can use the `chkptint`(1) utility to request either checkpointing at shutdown (interactive sessions only) or periodic checkpointing (either batch or interactive sessions).

Checkpointing is disabled by default. To allow users to checkpoint their sessions, the system administrator must set the URM configuration parameters that control checkpointing type (shutdown or periodic), frequency, interval type (CPU or clock), and length of time the checkpoint file is retained. See *UNICOS Resource Administration*, Cray Research publication SG–2302, for information on enabling checkpointing with `rmgr`.

Checkpointing is done on an individual session basis only. The `chkptint`(1) utility has one required option of the form `-s` *sec*, which specifies the requested checkpoint frequency in seconds. When URM is running in checkpoint-only-at-shutdown mode, all that is required is that `chkptint -s` be specified as a nonzero number. However, to enable periodic checkpointing, you must specify the interval in seconds as the *sec* argument to the `-s` option.

To turn off the automatic or periodic checkpoint request for that session, specify `chkptint -s 0`. The `chkptint`(1) utility can be made part of a script or placed in your `.cshrc` or `.profile` file.

The `view restart` subcommands apply to saved interactive sessions only. NQS jobs that use the `chkptint`(1) command are checkpointed and restarted by NQS, even though URM requested that the job be checkpointed at the appropriate time.

The `view restart` *nnn* subcommand works as follows:

```
rmgr-> view restart kcz
Restart images belonging to User <kcz>, UID 343 on path /ptmp/urm/chkpnt/kcz
        <1> 03201253.1520
        <2> 03211303.1520
        <3> 03231552.616
        <4> 03251613.616
        <5> 03281602.616
        <6> 03301623.616
User <kcz> has 6 restart images.
```

In the `view restart` *nnn n* subcommand, the *n* is the number in angle brackets (<>). This number is also used to specify which `chkpnt` file to restart and which `chkpnt` file to delete.

The form of the `chkpnt` file name is *MMddhhmm.nnnn*, that is, month(*MM*), day(*dd*), hour(*hh*), minute (*mm*), and the session ID of the job (*nnnn*).

To determine whether or not a restart file is a candidate for restart, use the `view restart` *nnn n* subcommand, which will display the processes and how much time has been used.

To dispose of `chkpnt` files you no longer need, use the `delete restart` *nnn n* subcommand. A `view restart` *nnn* or `view restart` *uuu* subcommand must be executed immediately before invoking the `delete` subcommand, as follows:

```
rmgr-> delete restart kcz 1
Deleted restart image <1> for user kcz
rmgr->
```

To restart a saved interactive session, first view the available `chkpnt` files by using the `view restart` *nnn* or `view restart` *uuu* subcommand, then, while in the same `rmgr` invocation, use the `restart` *nnn n* subcommand. If the restart is successful, the current session will be replaced by whatever was in the `chkpnt` file.

When a session is restarted, it resumes execution where it left off. For interactive sessions, this can sometimes be at an unexpected point, and the resulting output can be confusing. For example, if checkpointing occurred in the middle of the output of a `man`(1) command, the restarted session resumes output exactly where it left off. If an interactive session was checkpointed while waiting for input to a prompt, no prompt is displayed when the session is restarted.

See *UNICOS Resource Administration*, Cray Research publication SG–2302, for more information on checkpointing and restarting sessions.

NOTES

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
system, secadm, sysadm	Allowed to use this utility.

If the PRIV_SU configuration option is enabled, the super user is allowed to use this utility.

EXAMPLES

To review the current value for each URM object:

```
#rmgr
rmgr-> view /urm
LNr-r-  0    0 Sep  8 15:39 <machine       > -> node
LNrwr-  0    0 Sep  8 15:39 <Debug         > -> node
LVrwr-  0    0 Sep  8 15:39 <prevrun_boost > Float, 6.000000
LVrwr-  0    0 Sep  8 15:39 <entitle_wt    > Float, 0.500000
LVrwr-  0    0 Sep  8 15:39 <usage_wt      > Float, 0.500000
LVrwr-  0    0 Sep  8 15:39 <tape_wt       > Float, 0.500000
LVrwr-  0    0 Sep  8 15:39 <share_wt      > Float, 0.500000
LVrwr-  0    0 Sep  8 15:39 <service_wt    > Float, 0.500000
LVrwr-  0    0 Sep  8 15:39 <sds_wt        > Float, 0.500000
LVrwr-  0    0 Sep  8 15:39 <petime_wt     > Float, 0.500000
LVrwr-  0    0 Sep  8 15:39 <pe_wt         > Float, 0.500000
LVrwr-  0    0 Sep  8 15:39 <mem_wt        > Float, 0.500000
LVrwr-  0    0 Sep  8 15:39 <cpu_wt        > Float, 0.500000
LVrwr-  0    0 Sep  8 15:39 <bb_wt         > Float, 0.500000
LVrwr-  0    0 Sep  8 15:39 <age_wt        > Float, 0.500000
LVrwr-  0    0 Sep  8 15:39 <restart_switch>  "Force"
LVrwr-  0    0 Sep  8 15:39 <chkpnt_switch >  "No"
LVrwr-  0    0 Sep  8 15:39 <min_interval  >   Int, 1800
LVrwr-  0    0 Sep  8 15:39 <interval_type >  "Clock"
LVrwr-  0    0 Sep  8 15:39 <retain_chkpnt >   Int, 432000
LVr-r-  0    0 Sep  8 15:39 <shutdown      >   Int, 0
LVr-r-  0    0 Sep  8 15:39 <shut_done     >   Int, 0
LVrwr-  0    0 Sep  8 15:39 <restart_cmd   >  "UPATH/irstart"
LVrwr-  0    0 Sep  8 15:39 <chkpnt_path   >  "PPATH/chkpnt"
LVrwr-  0    0 Sep  8 15:39 <chkpnt_cmd    >  "UPATH/intchkpt"
LVrwr-  0    0 Sep  8 15:39 <sds_suspend   >  "UPATH/sdsspnd"
LVrwr-  0    0 Sep  8 15:39 <share_to_go   >   Int, 896
LVrwr-  0    0 Sep  8 15:39 <share_eval    >   Int, 900
LVrwr-  0    0 Sep  8 15:39 <timeout_user  >   Int, 600
LVrwr-  0    0 Sep  8 15:39 <sleep_time    >   Int, 10
LVrwr-  0    0 Sep  8 15:39 <sds_residence >   Int, 900
LVrwr-  0    0 Sep  8 15:39 <sched_delay   >   Int, 10
LVrwr-  0    0 Sep  8 15:39 <init_wait     >   Int, 1800
LVrwr-  0    0 Sep  8 15:39 <info_delay    >   Int, 10
```

```
LVrwr-  0    0 Sep  8 15:39 <share_policy  >  "Standard"
```

From within `rmgr`, to change the value of one of these URM objects (`sds_residence`, for example):

```
rmgr-> /urm/sds_residence = 100
```

SEE ALSO

`cd`(1), `chkptint`(1), `ustat`(1)

`urmsnap`(8), `urmd`(8), `usetjob`(8) in the *UNICOS Administrator Commands Reference Manual*, Cray Research publication SR–2022

UNICOS Resource Administration, Cray Research publication SG–2302

NAME

rpcgen – Generates code to implement Remote Procedure Call (RPC) protocol

SYNOPSIS

rpcgen *infile*
rpcgen [-c] [-h] [-l] [-m] [-o *outfile*] [*infile*]
rpcgen [-s *transport*] [-o *outfile*] [*infile*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The rpcgen compiler is a tool that generates C code to implement an RPC protocol. The input to rpcgen is a language with striking similarity to C, known as Remote Procedure Call Language (RPCL).

Typically, rpcgen is used as shown in the first synopsis, in which it takes an input file and generates four output files. If *infile* is named proto.x, rpcgen generates a header file in proto.h, xdr routines in proto_xdr.c, client-side stubs in proto_svc.c, and server-side stubs in proto_clnt.c.

When you do not want to generate all of the output files, but only a particular one, use the other synopses.

The input can contain C-style comments and preprocessor directives. Comments are ignored; the directives are simply stuffed uninterpreted into the output header file.

To customize eXternal Data Representation (XDR) routines, leave data types undefined. For every data type that is undefined, rpcgen assumes that a routine exists with xdr_ prepended to the name of the undefined type.

The rpcgen compiler accepts the following options:

infile	Specifies the input file.
-c	Compiles XDR routines.
-h	Compiles C data definitions (a header file).
-l	Compiles into client-side stubs.
-m	Compiles into server-side stubs, but does not generate a main routine. This option is useful for doing callback routines and for writing your own main routine to do initialization.
-o *outfile*	Specifies the name of the output file. If you omit *outfile*, standard output is used (-c, -h, and -s modes only).
-s *transport*	Compiles a server into server-side stubs by using the given transport. The supported transports are udp and tcp. You can invoke this option more than once to compile a server that serves multiple transports.

The following summary of RPCL syntax, which is used for `rpcgen` input, is to aid comprehension, rather than an exact statement of the language.

Primitive Data Types

RPCL primitive data types are as follows:

```
[unsigned] char
[unsigned] short
[unsigned] int
[unsigned] long
unsigned
float
double
void
bool
opaque
string
```

Except for the added `opaque`, `bool`, and `string` types, RPCL primitive data types are identical to those of C. The `rpcgen` compiler converts `bool` declarations to `int` declarations in the output header file (literally it is converted to `bool_t`, which has been defined through `#define` to be an `int`). `opaque` data types are converted to arrays of `chars`. You can declare `opaque` data as either a fixed- or variable-length array. C has no intrinsic `string` type, and it uses a null-terminated `char *` by convention. The RPCL keyword `string` is compiled into a `char *` in the output header file. Strings can be declared as having either a maximum length (for example, `string msg<80>;`) or an arbitrary length (for example, `string msg<>;`). Also, `void` declarations can appear only inside `union` and `program` definitions. Rather than typing the prefix `unsigned`, you can use abbreviations `u_char`, `u_short`, `u_int`, and `u_long`.

Declarations

RPCL allows only four kinds of declarations. Following is the format of each of these declarations:

Simple declaration: *type-name object-ident*

Pointer declaration: *type-name* `*`*object-ident*

Vector declaration: *type-name object-ident* `[`*size*`]`
The maximum *size* is specified between the angle brackets; you can omit *size*, indicating that the array can be of any size; *size* can be either an integer or a symbolic constant.

Variable-length-array declaration:

type-name variable-ident `<`*value*`>`
type-name variable-ident `< >`

Because variable-length arrays have no explicit syntax in C, these declarations are compiled into structures. For example, the declaration:

```
int heights <12>;
```

gets compiled into the following structure:

```
struct {

        u_int heights_len;

        int *heights_val;

} heights;
```

The number of items in the array is stored in the `_len` component, and the pointer to the array is stored in the `_val` component.

Type Definitions

The only way to generate an XDR routine is to define a type. For each type *zetype* you define, a corresponding XDR routine named *xdr_zetype* exists.

To define a type, you can use any of the following:

```
typedef
enumeration-def
structure-def
discriminated-union-def
program-def
const-def
```

The `typedef`, `enumeration-def`, and `structure-def` definitions are very similar to their C namesakes. C does not have a formal type mechanism to define variable-length arrays, and XDR unions are quite different from their C counterparts. Program definitions (`program-def`) are not type definitions in the same sense as the others, but they are useful nonetheless. The following bulleted paragraphs describe each of the type definitions and provide a syntax:

- `typedef`
  With RPCL, you cannot declare multidimensional arrays or pointers to pointers inline, unless you use `typedef`. The syntax for an XDR `typedef` is as follows:

  typedef:

  `typedef` *declaration* `;`

The *object-ident* part of *declaration* is the name of the new type; the *type-name* part is the name of the type from which it is derived.

- `enumeration-def`
  The syntax is as follows:

  `enum` *enum-ident* `{`
  enum-list
  `};`

  enum-list:
  enum-symbol-ident [= *assignment*]
  enum-symbol-ident [= *assignment*] , *enum-list*

The *assignment* variable can be either an integer or a symbolic constant. If there is no explicit assignment, the implicit assignment is the value of the previous enumeration plus 1. If not explicitly assigned, the first enumeration receives the value of 0.

- `structure-def`
  The syntax is as follows:

  `struct` *struct-ident* `{`
  declaration-list
  `};`

  declaration-list:
  declaration;
  declaration ; *declaration-list*

You cannot nest XDR definitions. For example, the following is an `rpcgen` error:

```
struct dontdoit {
        struct  ididit {
                int oops;
        } sorry;
        enum ididitagain { OOPS, WHOOPS } iapologize;
};
```

- `discriminated-union-def`
  The syntax is as follows:

union *union-ident* switch (*discriminant-declaration*) {
case-list
[default : *declaration* ;]

};

case-list:
case *case-ident* : *declaration* ;
case *case-ident* : *declaration* ; *case-list*

discriminant-declaration:
declaration

The union definition looks like a cross between a C-union and a C-switch. Following is an example:

```
union net_object switch (net_kind kind) {
case MACHINE:
        struct sockaddr_in sin;
case USER:
        int uid;
default:
        string whatisit;
};
```

The preceding example compiles into the following structure:

```
struct net_object {
        net_kind kind;
        union {
                struct sockaddr_in sin;
                int uid;
                char *whatisit;
        } net_object;
};
typedef struct net_object net_object;
```

The name of the union component of the output structure is the same as the name of the type itself.

- program-def

The syntax is as follows:

```
program program-ident {
        version-list
} = program-number ;

version-list:
        version
        version version-list

version:
        version version-ident {
                procedure-list
        } = version-number ;

procedure-list:
        procedure-declaration
        procedure-declaration procedure-list

procedure-declaration:
        type-name procedure-ident (type-name) = procedure-number ;
```

The following example shows a program definition. To create a server that can get or set the date, you can use the following declaration:

```
program DATE_PROG {
        version DATE_VERS {
                date DATE_GET(timezone) = 1;
                void DATE_SET(date) = 2;        /* Greenwich mean time */
        } = 1;
} = 100;
```

In the header file, this compiles into the following:

```
#define DATE_PROG 100
#define DATE_VERS 1
#define DATE_GET 1
#define DATE_SET 2
```

The client program should use these `define` statements to reference the remote procedures.

- `const-def`
  RPCL constants are symbolic constants that can be used wherever an integer constant is used. The syntax is as follows:

  `const` *const-ident* `=` *integer.*

An RPCL symbolic constant is used in the following array size specification:

```
const DOZEN = 12;
```

In the header file, this compiles into the following:

```
#define DOZEN 12
```

When using `rpcgen` to compile your server, the server interfaces to your local procedures by expecting a C function with the same name as that in the program definition, but it is in all lowercase letters and followed by the version number. The following is a local procedure that implements `DATE_GET`:

```
date *      /* always returns a pointer to the results */
date_get_1(tz)
    timezone *tz;       /* always takes a a pointer to the arguments */
{
        static date d;  /* must be static! */
        /*
         * figure out the date
         * and store it in d
         */
        return(&d);
}
```

XDR recursively frees the argument after getting the results from your local procedure; therefore, you should copy from the argument any data that you will need between calls. However, XDR neither allocates nor frees your results. You must handle their storage.

Inference Rules

You can set up suffix transformation rules in `make`(1) for compiling XDR routines, client and server-side stubs, and header files. The convention is that RPCL protocol files have the extension `.x`. An example of `make` rules to do this is as follows:

```
.SUFFIXES: .x
.x.c:
        rpcgen -c $< -o $@
.x.h:
        rpcgen -h $< -o $@
.x.l:
.x.m:
                .
                .
                .
```

BUGS

Name clashes can occur when you are using program definitions, because the apparent scoping does not really apply. You can avoid most of these by giving unique names for programs, versions, procedures, and types.

Nesting is not supported. As a workaround, you can declare at top-level, and use their name inside other structures to achieve the same effect.

SEE ALSO

Remote Procedure Call (RPC) Reference Manual, Cray Research publication SR–2089

NAME

rsv – Reserves tape resources

SYNOPSIS

rsv [-m *message_file*] [-t] [*resources*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The rsv utility reserves tape resources for you. Use rsv to reserve tape devices before opening any tape files. The rsv utility does not allocate real tape devices, but it gives you permission to access tape devices.

The rsv utility accepts the following options:

-m *message_file* Specifies a file in which informative messages from the tape subsystem are written. The default *message_file* is tape.msg, which the system administrator can change. If you omit *message_file*, the name defined during installation of the tape subsystem by MSGFILE in tapedef.h is used. The tape subsystem uses this file until all the reserved resources are released. The tape subsystem always appends to this file.

-t Places the message file in the directory specified by TMPDIR. See the CAUTIONS section. If the directory specified by TMPDIR does not exist or cannot be written into, the message file is created or used in the current working directory.

resources Takes the form *resource* [*amount*]; *resource* specifies the device group name, and *amount* specifies the number of devices. You can repeat *resource* [*amount*]; see the following examples. If you omit *resource*, it defaults to an installation-specified device group name. If you omit *amount*, one device is assumed.

Before you issue the rsv utility, you must release all previously reserved resources. If the requested resource is not available, or if you have not released all previously reserved resources, no new reservation occurs.

NOTES

If this utility is installed with a privilege assignment list (PAL), a user who is assigned the following privilege text upon execution of this command is allowed to perform the actions shown:

Privilege Text	Action
SMACDAC	Allowed to access any tape regardless of MAC and discretionary access control (DAC) restrictions. Device group MAC enforcement is still performed.
SDAC	Allowed to override DAC restrictions.

If this utility is installed with a PAL, a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
system, secadm	Allowed to read and write any tape, subject to security label restrictions on the device.
sysadm	Allowed to read and write any tape, subject to security label restrictions on the device and the tape.

If the PRIV_SU configuration option is enabled, root is allowed to override MAC and DAC restrictions. Device group MAC enforcement is still performed.

CAUTIONS

The file specified with the -m option or the default *message_file* in the current working directory is appended to and it may grow quite large.

The rsv utility creates a pipe in the directory defined as USER_DIR in tapedef.h; the default is $TMPDIR. If you change your $TMPDIR environment variable after a rsv utility, a tape request for tapeinfo fails.

EXIT STATUS

If rsv completes successfully, 0 is returned; otherwise, a nonzero value is returned. Where possible, this exit status code is normalized to the last three digits. Exit status values are documented in the *Tape Subsystem User's Guide*, Cray Research publication SG–2051.

EXAMPLES

The following examples illustrate different uses of the rsv utility.

Example 1: The rsv utility reserves one device named TAPE and one device named CART:

```
rsv TAPE 1 CART
```

Example 2: In the first command line of the example, rsv reads TAPE as a *resource* and 2 as the *amount*; it then reads CART as a *resource* and 1 as the *amount*. In the second command line, rsv reads TAPE as a *resource* and 1 as the *amount*; it then reads CART as the *resource*, and, because *amount* is not specified, one device is assumed.

```
rsv TAPE 2 CART 1
```

or

```
rsv TAPE 1 CART
```

Example 3: This example shows a usage of `rsv` that is not valid. In the first command line of the example, `rsv` reads `CART` as if it were an *amount*. In the second command line, `rsv` reads `2` as a *resource*.

```
rsv TAPE CART
```

or

```
rsv 2
```

Example 4: The `rsv` utility reserves one device of the default type, and it directs all messages to `msgfile` in your working directory:

```
rsv -m msgfile
```

If you move or remove the file specified by the `-m` option or the default *message_file* in the working directory before you are through processing tapes, messages will no longer be written.

FILES

`/usr/include/tapedef.h`	Definitions for trace file size

SEE ALSO

`privtext`(1), `rls`(1), `tpmnt`(1), `tprst`(1), `tpstat`(1)

General UNICOS System Administration, Cray Research publication SG–2301

Tape Subsystem User's Guide, Cray Research publication SG–2051

NAME

rusers – Lists names of users logged in on local machines (RPC version)

SYNOPSIS

/usr/bin/rusers [-ahilu] *hosts*

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The rusers utility produces output for remote machines that is similar to the output from the who(1) utility. It broadcasts on the local network and prints the responses it receives. (Cray Research systems are usually configured without connections to media that support broadcasting. Therefore, the rusers utility, without any hosts specified, may not return any user names.) Normally, the listing is in the order in which responses are received, but this order can be changed by the use of one of the options listed in this section.

The default is to print out a listing with one line per machine. When the -l flag is given, a who(1) style listing is used. If a user has not typed anything for a minute or more, the idle time is reported.

A remote host responds only if it is running the rusersd(8) daemon, which usually is started from inetd(8).

The rusers utility accepts the following options:

-a Displays a report for a machine even if no users are logged on.

-h Sorts alphabetically by host name.

-i Sorts by idle time.

-l Displays a longer listing in the style of who(1).

-u Sorts by number of users.

hosts When *host* arguments are given, rather than broadcasting, rusers will only query the list of specified hosts.

BUGS

Broadcasting does not work through gateways or on nonbroadcast media.

FILES

/etc/inetd.conf

SEE ALSO

`who`(1)

`inetd`(8), `rusersd`(8) in the *UNICOS Administrator Commands Reference Manual*, Cray Research publication SR–2022

NAME

`sact` – Prints current SCCS file-editing activity

SYNOPSIS

`sact` *files*

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The `sact` utility informs the user of any impending deltas to a named Source Code Control System (SCCS) file. This situation occurs when `get`(1) with the `-e` option has been executed previously without a subsequent execution of `delta`(1). If a directory is named on the command line, `sact` behaves as though each file in the directory were specified as a named file, except that non-SCCS files and unreadable files are silently ignored. If a name of `-` is given, the standard input is read with each line being taken as the name of an SCCS file to be processed.

The output for each named file consists of five fields separated by spaces, as follows:

Field 1 Specifies the SID of a delta that currently exists in the SCCS file to which changes will be made to make the new delta

Field 2 Specifies the SID for the new delta to be created

Field 3 Contains the log name of the user who will make the delta (that is, executed a `get` for editing)

Field 4 Contains the date that `get -e` was executed

Field 5 Contains the time that `get -e` was executed

EXIT STATUS

The `sact` utility exits with one of the following values:

0 Successful completion.

>0 An error occurred.

MESSAGES

Error messages from SCCS are printed. Use `help`(1) for explanations.

SEE ALSO

admin(1), cdc(1), comb(1), delta(1), get(1), help(1), prs(1), rmdel(1), sccsdiff(1), unget(1), val(1), vc(1), what(1)

sccsfile(5) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

NAME

sag – Displays system activity graph

SYNOPSIS

sag [-s *time*] [-e *time*] [-i *sec*] [-f *file*] [-T *term*] [-x *spec*] [-y *spec*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The sag command graphically displays the system activity data stored in a binary data file by a previous sar(1) run. You can plot any of the sar(1) data items singly or in combination, as crossplots or versus time; simple arithmetic data combinations can be specified. The sag command invokes sar(1) and finds the desired data by string-matching the data column header (run sar to see what is available). The following options are passed to sar(1):

-s *time* Selects data later than *time* in the form *hh*[:*mm*]. The default is 08:00.

-e *time* Selects data up to *time*. The default is 18:00.

-i *sec* Selects data at intervals as close as possible to *sec* seconds.

-f *file* Uses *file* as the data source for sar(1). The default is the current daily data file /usr/adm/sa/sa*dd*.

The following options are also available:

-T *term* Produces output suitable for terminal *term*.

-x *spec* Produces x axis with *spec* in the form shown under the -y option.

-y *spec* Produces y axis with *spec* in the following form:

"*name*[*op name*]...[*lo hi*]"

The *name* variable is either a string that matches a column header in the sar report, with an optional device name in brackets (for example, reads[dsk-1]) or an integer value. *op* is the symbol +, -, ∗, or / surrounded by blanks. You can specify up to five names. sag does not recognize parentheses. Contrary to custom, + and - have precedence over ∗ and . Evaluation is left to right. Thus, A / A + B * 100 is evaluated (A/(A+B))∗100, and A + B / C + D is (A+B)/(C+D). *lo* and *hi* are optional numeric scale limits. If unspecified, they are deduced from the data.

A single *spec* is permitted for the x axis. If unspecified, *time* is used. Up to 5 *spec*s separated by semicolons (;) may be given for -y. Enclose the -x and -y arguments in double quotes ("") if blanks or \<CR> are included. The -y default is as follows:

```
-y "%usr 0 100; %usr + %sys 0 100; %usr + %sys + %wio 0 100"
```

BUGS

The `sag` command produces a command file consisting of graphic commands and several data files. These must be used on a machine that supports plots or graphs, such as a Sun Workstation. The files are transformed to graphs on a Sun workstation by using `tektool`, then executing the `sag`-produced command file, and piping the output through plot. The scaling produced by `sag` is not correct for Cray PVP systems.

EXAMPLES

Example 1: To see today's CPU usage, enter the following:

```
sag
```

Example 2: To see activity over 15 minutes of all disk drives, enter the following:

```
TS=`date +%H:%M`
sar -o tempfile 60 15
TE=`date +%H:%M`
sag -f tempfile -s $TS -e $TE -y "reads"
```

FILES

`/usr/adm/sa/sa`*dd* Daily data file for day *dd*

SEE ALSO

`sar`(1)

NAME

sar – Extracts operating system activity information

SYNOPSIS

sar [-a] [-b] [-c] [-d] [-g] [-h] [-j] [-k] [-l] [-n] [-o *file*] [-p] [-q] [-r] [-t] [-u] [-v] [-w] [-x] [-y] [-z] [-A] [-B] [-H] [-L] [-M] [-P] [-S] [-T] [-U] [-W] [-X] [-Z] *seconds* [*integral*]

sar [-a] [-b] [-c] [-d] [-e *time*] [-f *file*] [-g] [-h] [-i *sec*] [-j] [-k] [-l] [-n] [-p] [-q] [-r] [-s *time*] [-t] [-u] [-v] [-w] [-x] [-y] [-z] [-A] [-B] [-H] [-L] [-M] [-P] [-S] [-T] [-U] [-W] [-X] [-Z]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The sar command extracts operating system activity information according to a specified time interval. In the first instance, sar samples cumulative activity counters in the operating system. You specify the number of seconds between samples (*seconds*) and the total number of samples (*integral*). If you specify -o *file*, sar saves the samples in *file* in binary format.

In the second instance, sar extracts data from a previously recorded *file*, either the one specified by the -f option or, by default, the standard system activity daily data file, /usr/adm/sa/sa*dd*, for the current day *dd*. The starting and ending times of the report can be bounded through the -s *time* and -e *time* arguments by using the 24-hour clock form *hh*[:*mm*[:*ss*]]. The -i option selects records at *sec* second intervals; otherwise, all intervals found in the data file are reported.

In either case, you can specify subsets of data to be printed by using the following options (the fields reported are listed with each option):

-a	Reports use of file access system routines.	
	iget/s	Number of inode accesses per second.
	namei/s	Number of file path name lookups per second.
	dirblk/s	Number of file directory blocks read per second.
-b	Reports buffer activity.	
	bread/s, bwrit/s	Transfers per second of data between system buffers and disk or other block devices.
	lread/s, lwrit/s	Accesses of system buffers.
	%rcache, %wcache	Percentage of read/write buffer accesses that were statisfied by the buffer cache instead of having to go to a disk directly.
	pread/s, pwrit/s	Transfers through raw (physical) device mechanism.
-c	Reports system calls.	

`scall/s` System calls of all types.
`sread/s`, `swrit/s`, `fork/s`, `exec/s`
Specific system calls.
`rchar/s`, `wchar/s` Characters transferred by read and write system calls.

`-d` Reports activity for each disk device.

`reads writes` Number of data transfers from or to device during the interval. The number of bytes transferred can be computed by multiplying reads+writes by 4096.

On systems with an I/O subsystem model E (IOS-E):

`reads writes` Number of blocks transferred.
`ncyls` Number of cylinders crossed.
`avwait`, `avserv` Average time in milliseconds that transfer requests wait idly in the queue and average time to be serviced. This includes seeks, rotational latency, and data transfer times.
`rerrs`, `uerrs` Number of total recovered and unrecovered read and writer errors.

`-e` *time* Specifies the end time of report.

`-f` *file* Specifies the *file* from which `sar` extracts data.

`-g` Reports host kernel calls issued by a guest kernel. The information is displayed analogously to the `-t` option and includes the following:

`host call` The host call name.
`%time` Time spent by the host handling each call type as a percent of the interval total.
`calls/s` Number of calls per second.
`avetime` Average handling time (in microseconds).
`maxtime` Maximum handling time (in microseconds) across all reporting intervals.
`mintime` Minimum handling time (in microseconds) across all reporting intervals.

`-h` Reports terminal traffic and possible overflow of terminal (`clist`) buffers.

`-i` *sec* Selects records at *sec* intervals.

`-j` Reports process shuffles in memory and text/data locking and unlocking. Process shuffles occur when a `plock`(2) or `chmem`(2) system call is initiated or when real time mode is set. Text/data locking and unlocking occurs when `plock` is called.

`-k` Reports TCP/IP interrupt information. The system and the `sar` package must have been built with SCTRACE defined, and TCP/IP must be in the system.

`-l` Reports device cache (`ldcache`) activity.

`Cache to user: Reads`, `Writes`
Number of blocks read from cache to user and number of blocks written from user to cache.

`Cache to disk: Reads`, `Writes`
: Number of blocks read from disk to cache and number of blocks written from cache to disk.

`Cache/disk ratio: Read`, `Write`, `Total`
: Ratio of cache-to-user to cache-to-disk.

The `-l` option is only available to non-root users when `sar` is extracting data from a previously recorded file. It is not available to non-root users when `sar` is collecting data interactively from the system (when the seconds and integral parameters are specified), because the device from which these statistics are gathered is accessible only by root.

`-n` Reports network activity.

`network` Network address.

`ipkts/s`, `ierrs/s`, `opkts/s`, `oerrs/s`, `collis/s`
: Corresponds to input packets, input errors, output packets, output errors, and collisions, respectively.

`-o` *file* Saves the samples in *file* in binary format.

`-p` Reports CPU usage by processor.

`unix restarts: cpu`, `user`, `unix`, `idle`
: Portion of time running in CPU mode, user mode, running in the kernel, and otherwise idle, respectively.

`-q` Reports average queue length while occupied, and percentage of time occupied.

`runq-sz`, `%runocc` Run queue of processes in memory that are runnable.
`swpq-sz`, `%swpocc` Swap queue of processes swapped out, but ready to run.

`-r` Reports remote file (network file system (NFS)) activity.

`svcall/s` Server calls per second.
`%svread` Percent server reads.
`%svwrit` Percent server writes.
`%svother` Percent all others.
`clcall/s` Client calls per second.
`%clread` Percent client reads.
`%clwrite` Percent client writes.
`%clother` Percent all others.

`-s` *time* Specifies the start time of the report.

`-t` Reports system call information. This includes the system call name, percent of time spent in the call, number of calls per second, and the average, minimum, and maximum system call path lengths. Unused system calls during the period of time being reported are not shown. The system and the `sar` package must have been built with SCTRACE defined.

`-u` Reports CPU usage. The default report contains the following fields:

%usr	Portion of time running in user mode.
%sys	Portion of time running in system mode.
%wsem	Portion of time waiting on a semaphore.
#locks	Number of collisions for the system lock.
%idle	Time spent idle.
%wio	Time spent idle waiting for I/O.
%guest	Portion of time used by the guest(s). If sar is being run from a guest, the time reported is for the host.

%usr, %sys, %idle, and %guest make up the total time. %wsem is a component of %sys. %wio is a component of %idle.

-v Reports status of text, process, nc1inode, and file tables.

text-sz, proc-sz, nc1inod-sz, file-sz
: Entries/size for each table, evaluated once at sampling point.

text-ov, proc-ov, nc1inod-ov, file-ov
: Overflows occurring between sampling points.

-w Reports system swapping and switching activity.

swpin/s, swpot/s, bswin/s, bswot/s	Number of transfers and number of 4096-byte units transferred for swapins (including initial loading of some programs) and swapouts.
xswin/s, xswot/s	Number of times a shared-text process was swapped in and the number of times a shared-text segment was freed.
pswch/s	Process switches.

-x Reports IOS packets into the Cray Research system and out to the IOS.

-y Reports tty device activity.

rawch/s, canch/s, outch/s	Input character rate, input character rate processed by canon, and output character rate.
rcvin/s, xmtin/s	T-packets received and transmitted (a t-packet is a terminal packet type).

-z Reports asynchronous I/O usage; in other words, sysrda and syswra report buffer cache asynchronous usage. aread, awrite, and listio report usage of reada, writea, and listio system calls.

-A Reports all data. Equivalent to specifying -abcdhjklpqtuvwxyzBHMTXWZ.

-B Produces an expanded version of the -b report. In addition to the information found in the -b report, the following additional information is given:

brblks, bwblks	Number of blocks transferred (read/written) between system buffers and disk or other block devices.
lrblks, lwblks	Number of blocks moved from/to system buffers.
prblks, pwblks	Number of blocks read/written via raw (physical) I/O.

`-H` System call history option. Reports system call information for the time period from boot to the present. This information includes the system call name, percent of time spent in the call, number of calls per second, and the average, minimum, and maximum system call path lengths. Unused system calls during the period being reported are not shown. The system and the `sar` package must have been built with SCTRACE defined.

`-L` Reports kernel multi-threaded lock statistics.

`Lid`	Kernel lock identifier.
`locks/s`	Number of lock attempts per second.
`avethrd`	Average kernel thread time in microseconds while locked.
`hold/s`	Number of lock attempts per second that resulted in a hold to wait while another CPU had the lock.
`avehold`	Average hold time in microseconds.
`%wsem`	Percent of wait time this kernel lock accounted for.

NOTE: For data to appear in the `locks/s` and `avethrd` fields, the kernel must have been built using the `SEMTIMING` option. By default, this is not done, due to the significant kernel overhead that it adds.

`-M` Reports memory and swap usage.

`umemtot`	Amount of memory available for use by user processes.
`umemuse`	Amount of user memory in use.
`memlock`	Amount of locked memory.
`swaptot`	Total amount of swap space.
`swapuse`	Amount of swap space in use.

`-P` Reports the same data as the `-p` option, except the `unix` field has been expanded into two fields.

`unixc`	Percent of time running in system mode executing system calls on behalf of users.
`unixk`	Percent of time running in system mode executing kernel functions on behalf of the system. This includes interrupt processing time and semaphore wait time.

`-S` Summary format. Reports raw totals from the last `unix restart` or from a specified interval. Valid only for the `-b`, `-d`, `-n`, `-r`, `-t`, `-u`, `-w`, and `-L` options.

`-T` Reports activity for each tape drive.

`mounts`	Number of volumes mounted.
`reads writes`	Number of data transfers from or to the device during the interval. The number of bytes transferred can be computed by multiplying each value by 4096.

`-U` Reports the same data as the `-u` option, except the `sys` field has been expanded into two fields, not including semaphore wait time.

`sysc`	Percent of time running in system mode executing system calls on behalf of users.

`sysk` Percent of time running in system mode executing kernel functions on behalf of the system. This includes interrupt processing time.

`-W` Reports the number of times per second that processes are loaded and runnable, but not running, and the average number of processes.

`-X` Reports the number of abnormal exchanges from user programs, by CPU (abnormal exchanges = err, fpi, ore, pre, dli).

`-Z` Reports the same data as the `-z` option, plus the number of blocks transferred by `aread` and `awrite`.

seconds Specifies the number of seconds between samples.

[*integral*] Specifies the total number of samples.

CAUTIONS

When collecting `sar` data over short intervals (a few seconds), it is possible for `sar` to report values that appear inconsistent. This is true when using `sar` to collect the data or by calling `sadc` directly (see `sar`(8)). For example, the percentage displayed in the `%swpocc` field is more than 100%. This occurs because not all the data values used by `sar` are updated at exactly the same rate. In addition, `sadc` samples data in several steps, which can lead to additional inaccuracies if updates occur between sampling steps. Consider the `%swpocc` field. This field is calculated by subtracting the old data from the new data and dividing by time; however, since the data is only updated 1 time per second by the kernel, it is likely that data samples taken at a 1-second rate will have percentage values greater than 100%.

EXAMPLES

Example 1: The following example shows today's CPU activity so far. Data is extracted from the file `/usr/adm/sa/sa`*dd*.

```
$ sar
```

Example 2: The following example collects three samples that contain data about all system activity. Data samples are written to the `tmp` file every 20 minutes.

```
$ sar -o temp 1200 3
```

Example 3: The following example reports disk and tape activity for data collected in example 2.

```
$ sar -dT -f temp
```

FILES

`/usr/adm/sa/sa`*dd* Daily data file, where *dd* are digits representing the day of the month

SEE ALSO

`sag`(1)

`chmem`(2), `plock`(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

`sar`(8) in the *UNICOS Administrator Commands Reference Manual*, Cray Research publication SR–2022

NAME

scanit – Corrects code for certain user programs on CRAY J90 systems and CRAY EL98 systems

SYNOPSIS

scanit [-e *feature*] [-o *output_file*] [-v] *input_file*

IMPLEMENTATION

Cray PVP systems

Necessary for multitasking code executing on CRAY EL98 systems and code that enables scalar cache on CRAY J90 systems.

DESCRIPTION

Hardware instruction sequences have been discovered that can generate unexpected results in certain situations. segldr(1) and the scanit command render an executable file (a.out) safe in the following situations:

- When it uses multitasking on CRAY EL98 systems
- When it enables cache memory on CRAY J90 systems

If your program does not fall into one of these categories, or if you load your program with SEGLDR version 8.0.4 or later, you have no reason to use scanit. The segldr(1) command and scanit provide the same safety features.

scanit first analyzes and then modifies your a.out file. It addresses only the two issues it was designed to make safe; it will not affect the program's performance significantly or change the answers you receive.

scanit accepts the following options:

-e *feature* Analyzes and makes safe the program that makes use of *feature*. The choices for *feature* are as follows:

- cache — Makes *output_file* safe for using cache on CRAY J90 systems.
- tasking — Makes *output_file* safe for multitasking on CRAY EL98 systems.

-o *output_file* Names the file output by scanit. The default name is a.out; however, scanit will not overwrite *input_file*, so ensure that the two names do not match.

-v Prints additional analysis information: specifically, the number of instruction sequences containing a possible hazard. Use this information to determine whether a hazard condition exists.

input_file Names the executable program to be analyzed and made safe.

ENVIRONMENT VARIABLES

If you set the TARGET environment variable to identify the machine characteristics of the system on which the program will execute, scanit will use that information. If you do not use the TARGET variable, scanit assumes the host system. The use of the -e *feature* option overrides the TARGET environment variable or host system machine characteristics.

NOTES

You are strongly urged to reload your CRAY EL98 and CRAY J90 programs with SEGLDR version 8.0.4 or later. The scanit command should only be used when reloading with segldr(1) is impractical. While the behavior of scanit mimics that of segldr(1), scanit operates on an existing executable program rather than .o files and libraries. segldr(1) contains no new options to make programs safe; it performs the analysis and any corrections automatically, based on the TARGET information for the host system.

In rare circumstances, you may be unable to reload a program with segldr(1). For instance, this is the case when the source files for a program are not available because the program is distributed in executable form by a vendor other than Cray Research. If you have an agreement with another vendor that does not allow you to modify executable code, do not use scanit.

When the scanit command produces an executable program that can safely use cache on a CRAY J90 system, it sets a bit in the output a.out file to indicate to the UNICOS kernel that cache should be enabled.

Be aware that the scanit tool may have difficulty finding enough fixup space in segmented programs if the segments are small. Relinking segmented codes with SEGLDR version 8.0.4 or later should work, because it adds extra code space in non-root segments.

EXIT STATUS

The scanit command exits with one of the following values:

0 Successful completion. The output file can be safely executed.

1 The scanit command was unable to make the output file safe. A warning message to this effect will be printed to stderr. In addition, if run it on a CRAY J90 system, the program will run without cache enabled (the enable-cache bit will not be set). On a CRAY EL98 system, the permission bits to enable execution will not be set. If you change the permission bits and run the program on a CRAY EL98 system, you may get error conditions or incorrect answers; the program will run correctly on other Cray PVP systems.

EXAMPLES

Example 1: In the following example, all of the defaults are used. scanit uses TARGET information to determine whether the host is a CRAY J90 or a CRAY EL98 system (that is, whether to fix cache access or multitasking problems) and writes the output to a.out.

```
scanit my.code
exec a.out
```

Example 2: Use the following command line on a CRAY EL98 multitasking executable named `a.out`:

```
scanit -e tasking -o safe.out a.out
exec safe.out
```

Example 3: Use the following command line to access cache memory on a CRAY J90 system. This example takes an executable file named `program.output` and, because the `-o` option is missing, writes the output to `a.out`, the default. The `-v` option prints the number of problems corrected.

```
scanit -v -e cache program.output
File a.out contains fixes for 628 cache hazard conditions

exec a.out
```

If there had been no problems with the executable file, the `-v` option would have printed the following:

```
No hazard conditions were found in file program.output
```

SEE ALSO

`segldr`(1)

NAME

sccs – Front end for the SCCS subsystem

SYNOPSIS

sccs [-r] [-d *path*] [-p *path*] *command* [*options...*] [*operands...*]

IMPLEMENTATION

All Cray Research systems

STANDARDS

XPG4

DESCRIPTION

The sccs utility is a front end to the SCCS programs. It also includes the capability to run set-user-id to another user to provide additional protection.

The sccs utility invokes the specified *command* with the specified *options* and *operands*. By default, each of the *operands* is modified by prefixing it with the string SCCS/s.

The *command* operand can be one of the SCCS utilities in this document (admin, delta, get, prs, rmdel, sact, unget, val, or what) or one of the pseudo-utilities listed in the DESCRIPTION section.

The sccs utility accepts the following options, except that *options* operands are actually options to be passed to the utility named by *command*. When the portion of the command:

command [*options...*] [*operands...*]

is considered, all of the pseudo-utilities used as *command* support the Utility Syntax Guidelines. Any of the other SCCS utilities that can be invoked in this manner support the Guidelines to the extent indicated by their individual OPTIONS sections.

The following options are supported preceding the *command* operand:

-d *path* A path name of a directory to be used as a root directory for the SCCS files. The default is the current directory. The -d option takes precedence over the PROJECTDIR variable. See -p.

-p *path* A path name of a directory in which the SCCS files are located. The default is the SCCS directory.

The -p options differs from the -d option in that the -d option-argument is prefixed to the entire path name and the -p option-argument is inserted before the final component of the path name. For example:

```
sccs -d /x -p y get a/b
```

will convert to:

```
get /x/a/y/s.b
```

This allows the creation of aliases such as:

```
alias syssccs="sccs -d /usr/src"
```

that will be used as:

```
syssccs get cmd/who.c
```

`-r` Invokes *command* with the real user ID of the process, not any effective user ID that the `sccs` utility is set to. Certain commands (`admin`, `check`, `clean`, `diffs`, `info`, `rmdel`, and `tell`) cannot be run set-user-ID by all users, since this would allow anyone to change the authorizations. These commands are always run as the real user.

The `sccs` utility accepts the following operands:

command An SCCS utility name or the name of one of the pseudo-utilities listed in the DESCRIPTION section.

options An option or option-argument to be passed to *command*.

operands An operand to be passed to *command*.

The following environment variables affect the execution of `sccs`:

`LANG` Provides a default value for the internationalization variables that are unset or null. If `LANG` is unset or null, the corresponding value from the implementation-specific default locale will be used. If any of the internationalization variables contains an invalid setting, the utility will behave as if none of the variables had been defined.

`LC_ALL` If set to a nonempty string value, override the values of all the other internationalization variables.

`LC_CTYPE` Determines the locale for the interpretation of sequences of bytes of text data as characters (for example, single- as opposed to multibyte characters in arguments and input files).

`LC_MESSAGES` Determines the locale that should be used to affect the format and contents of diagnostic messages written to standard error.

`NLSPATH` Determines the location of message catalogs for the processing of `LC_MESSAGES`.

`PROJECTDIR` Provides a default value for the `-d` *path* option. If the value of `PROJECTDIR` begins with a slash, it is considered an obsolete path name; otherwise, the home directory of a user of that name is examined for a subdirectory `src` or `source`. If such a directory is found, it is used. Otherwise, the value is used as a relative path name.

Additional environment variable effects may be found in the utility description for the specified *command*.

Many of the SCCS utilities take directory names as operands as well as specific filenames. The pseudo-utilities supported by `sccs` are not described as having this capability, but are not prohibited from doing so.

The following pseudo-utilities are supported as *command* operands. All options referred to in the following list are values given in the *options* operands following *command*.

`check` Equivalent to `info`, except that nothing is printed if nothing is being edited, and a non-zero exit status is returned if anything is being edited. The intent is to have this included in an "install" entry in a makefile to ensure that everything is included into the SCCS file before a version is installed.

`clean` Removes everything from the current directory that can be recreated from SCCS files, but do not remove any files being edited. If the `-b` option is given, branches are ignored in the determination of whether they are being edited; this is dangerous if branches are kept in the same directory.

`create` Creates an SCCS file, taking the initial contents from the file of the same name. Any options to `admin` are accepted. If the creation is successful, the original files are renamed by prefixing the basenames with a comma. These renamed files should be removed after it has been verified that the SCCS files have been created successfully.

`delget` Performs a `delta` on the named files and then `get` new versions. The new versions will have ID keywords expanded and will not be editable. Any `-m`, `-p`, `-r`, `-s`, and `-y` options will be passed to `delta`, and any `-b`, `-c`, `-e`, `-i`, `-k`, `-l`, `-s`, and `-x` options will be passed to `get`.

`deledit` Equivalent to `delget`, except that the `get` phase includes the `-e` option. This option is useful for making a checkpoint of the current editing phase. The same options will be passed to `delta` as described above, and all the options listed for `get` above except `-e` are passed to `edit`.

`diffs` Writes a difference listing between the current version of the files checked out for editing and the versions in SCCS format. Any `-r`, `-c`, `-i`, `-x`, and `-t` options are passed to `get`; any `-l`, `-s`, `-e`, `-f`, `-h`, and `-b` options are passed to `diff`. A `-C` option is passed to `diff` as `-c`.

`edit` Equivalent to `get -e`.

`fix` Removes the named delta, but leaves a copy of the delta with the changes that were in it. It is useful for fixing small compiler bugs, and so forth. It must be followed by a `-r` *SID* option. Since `fix` does not leave audit trails, it should be used carefully.

`info` Writes a listing of all files being edited. If the `-b` option is given, branches (that is, SIDs with two or fewer components) are ignored. If a `-u` *user* option is given, then only files being edited by the named user are listed. A `-U` option is equivalent to `-u` <*current user*>.

`print` Writes out verbose information about the named files, equivalent to `sccs prs`.

`print` Writes a newline-separated list of the files being edited to standard output. Takes the `-b`, `-u`, and `-U` options like `info` and `check`.

`unedit` This is the opposite of an `edit` or a `get -e`. It should be used with caution, since any changes made since the `get` will be lost.

EXIT STATUS

The `sccs` utility exits with one of the following values:

0 Successful completion.

>0 An error occurred.

EXAMPLES

Example 1: To get a file for editing, edit it and produce a new delta:

```
sccs get -e file.c
ex file.c
sccs delta file.c
```

Example 2: To get a file from another directory:

```
sccs -p /usr/src/sccs/s. get cc.c
```

or

```
sccs get /usr/src/sccs/s.cc.c
```

Example 3: To make a delta of a large number of files in the current directory:

```
sccs delta *.c
```

Example 4: To get a list of files being edited that are not on branches:

```
sccs info -b
```

Example 5: To delta everything being edited by the current user:

```
sccs delta $(sccs tell -U)
```

Example 6: In a makefile, to get source files from an SCCS file if it does not already exist:

```
SRCS = <list of source files>
$(SRCS):
       sccs get $(REL) $@
```

NAME

`sccsdiff` – Compares two versions of an SCCS file

SYNOPSIS

`sccsdiff -r`*SID1* `-r`*SID2* [`-p`] [`-s`*n*] *files*

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `sccsdiff` utility compares two versions of a Source Code Control System (SCCS) file and generates the differences between the two versions. Any number of SCCS files may be specified; the arguments apply to all files.

The `sccsdiff` utility accepts the following options and arguments:

`-r`*SID#* *SID1* and *SID2* specify the deltas of an SCCS file that are to be compared. Versions are passed to `bdiff`(1) in the order given.

`-p` Pipes output for each file through `pr`(1).

`-s`*n* *n* is the file segment size that `bdiff`(1) passes to `diff`(1). This is useful when `diff`(1) fails because of a high system load.

files Specifies the SCCS files to be compared.

MESSAGES

`file: No differences` The two versions are the same.

Error messages from SCCS are printed. Use `help`(1) for explanations.

EXAMPLES

The differences between delta `1.1` and `1.2` in file `s.example.c` are written to `stdout`. User input is shown in bold type:

```
$ sccsdiff -r1.1 -r1.2 s.example.c
4c4
<       printf("Hello, world\n");
---
>       printf("Hello, world!\n");
$
```

FILES

`/tmp/get`*?????* Temporary files

SEE ALSO

`admin`(1), `bdiff`(1), `cdc`(1), `comb`(1), `delta`(1), `diff`(1), `get`(1), `help`(1), `pr`(1), `prs`(1), `rmdel`(1), `sact`(1), `unget`(1), `val`(1), `vc`(1), `what`(1)

`sccsfile`(5) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

NAME

`script` – Makes a typescript of a terminal session

SYNOPSIS

`script` [`-a`] [`-k`] [`-n`] [`-q`] [`-s`] [`-S` *shell*] [*file*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `script` utility saves characters written to your terminal in a file. If you do not specify a *file* name, the characters are saved in a file called `typescript`.

The script ends when the forked *shell* exits.

This program is useful when you are using a CRT and want a copy of the dialog.

The `-k`, `-n`, `-s`, and `-S` options control which shell is used. If these options are not specified, `script` will attempt to determine the correct shell from the environment.

The `script` utility accepts the following options:

`-a` Appends to the `typescript` file instead of creating a new file.

`-k` Invokes `/bin/sh`.

`-n` Invokes `/bin/csh`.

`-q` Invokes `quiet mode`, in which the `script started` and `script done` messages are turned off.

`-s` Invokes `/bin/sh`.

`-S` *shell* Lets you specify the shell.

file File that collects characters written to your controlling terminal. The default depends on your `SHELL` environment variable.

SEE ALSO

`chown`(2), `select`(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

`pty`(4) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

NAME

`sdiff` – Compares programs side-by-side

SYNOPSIS

`sdiff` [`-l`] [`-o` *output*] [`-s`] [`-w` *n*] *file1 file2*

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `sdiff` utility uses the output of `diff`(1) to produce a side-by-side listing of two files, indicating the lines that are different. Each line of the two files is printed with a blank gutter between them if the lines are identical, a `<` in the gutter if the line exists only in *file1*, a `>` symbol in the gutter if the line exists only in *file2*, and a `|` symbol for lines that are different.

Example:

```
x       |       y
a               a
b       <
c       <
d               d
        >       c
```

The `sdiff` utility accepts the following options:

`-l` Prints only the left side of any lines that are identical.

`-o` *output* Uses the next argument, *output*, as the name of a third file that is created as a user-controlled merging of *file1* and *file2*. Identical lines of *file1* and *file2* are copied to *output*. Sets of differences, as produced by `diff`(1), are printed; where a set of differences share a common gutter character. After printing each set of differences, `sdiff` prompts the user with % and waits for one of the following user-typed commands:

- `l` Appends the left column to the output file.
- `r` Appends the right column to the output file.
- `s` Turns on silent mode; does not print identical lines.
- `v` Turns off silent mode.
- `e l` Calls the editor with the left column.
- `e r` Calls the editor with the right column.
- `e b` Calls the editor with the concatenation of left and right.
- `e` Calls the editor with a zero-length file.

`q` Exits from the program.

On exit from the editor, the resulting file is concatenated on the end of *output.*

`-s` Does not print identical lines.

`-w` *n* Uses the next argument, *n*, as the width of the output line. The default line length is 130 characters.

file1, *file2* Files to be compared.

NOTES

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
`system`, `secadm`	Allowed to compare any two files. In a privileged administrator shell environment, shell-redirected I/O is not subject to file protections.
`sysadm`	Allowed to compare any two files subject to security label restrictions. Shell-redirected I/O is subject to security label restrictions.

If the `PRIV_SU` configuration option is enabled, the super user is allowed to compare any two files. Shell-redirected I/O on behalf of the super user is not subject to file protections.

CAUTIONS

Lines from input files which exceed `sdiff`'s internal buffer are truncated.

SEE ALSO

`diff`(1), `ed`(1)

NAME

sdss – Reports status information about the secondary data segment pool

SYNOPSIS

sdss [-a] [-c] [-d] [-e] [-l] [-r] [-s] [-w] [-j *jidlist*] [-p *pidlist*] [-u *uidlist*]

IMPLEMENTATION

Cray PVP systems (except CRAY J90 series and CRAY EL series)

DESCRIPTION

The sdss command prints information about processes and logical device caches by using the secondary data segments (SDS) pool. The following options control the information display:

-a Prints the base address of the SDS. Must be accompanied by the -c or -d option.

-c Prints information about logical device cache segments.

-d Prints information about all processes with existing SDS.

-e Implies all other options.

-l Prints information about SDS limits. Implies the -d option.

-r Raw mode; suppresses the header and summary lines. This is useful for piping the output into sort(1).

-s Prints the login name, rather than the numerical user ID. Implies the -d option.

-w Prints information about processes waiting on an ssbreak(2) system call. Implies the -d option.

-j *jidlist* Restricts the listing to data about processes that have job ID numbers in *jidlist*. Implies the -d option.

-p *pidlist* Restricts the listing to data about processes that have process ID numbers in *pidlist*. Implies the -d option.

-u *uidlist* Lists only data about processes that have a user ID number or login name in *uidlist*. Implies the -d option.

If you do not specify any options, an information summary is given detailing basic usage statistics.

Definitions for the column headings in an sdss listing follow. The letters under the option heading indicate the options that cause the corresponding heading to appear. Note that a ! symbol preceding an option letter indicates that the option has not been specified.

Heading	Option	Description
ADDR	-a	Base address of the segment relative to the base address of the SDS pool.
SIZE	-c,-d	Size of the segment in SDS units (a unit is 4096 bytes).
SBRK	-d,-w	Size of the ssbreak increment for which the process is waiting.
LIM	-d,-l	Maximum SDS size allowed to the process. A letter J appended to the value indicates a per job limit, and a letter P appended to the value indicates a per process limit.
PID	-d	Process ID of the process; you can kill or checkpoint a process if you know this number.
JID	-d	Job ID of the process; you can checkpoint a job if you know this number.
UID	-d	User ID number of the process owner; the login name is printed under the -s option.
F	!-l,!-w	Flags (octal and additive) associated with the process: 01 In core 02 System process 04 Locked in core (as for physical I/O) 10 Being swapped 20 Being traced by another process 100 Connected to CPU 200 Suspended for single threading 2000 Suspended for deadlock 4000 Suspended by user 10000 CPU limit exceeded 20000 Recoverable process 40000 Selecting 100000 Idle process 200000 Suspend in process 400000 Another tracing flag

Heading	Option	Description
S	!-l,!-w	The state of the process: S Sleeping W Waiting R Running 0 Running, connected to CPU 0 1 Running, connected to CPU 1 2 Running, connected to CPU 2 3 Running, connected to CPU 3 I Intermediate Z Terminated T Stopped X Growing
PRI	!-l&!-w	Priority of the process; higher numbers mean lower priority.
NI	!-l&!-w	Nice value; used in priority computation.
TIME	-d	Cumulative execution time for the process.
CMD/LDCACHE	-c,d	Command name or the logical device cache name.

NOTES

Output from sdss is restricted to processes running at a security label that the calling user dominates.

If this command is installed with the default privilege assignment list (PAL), a user with the showall privilege text is not subject to output restrictions.

BUGS

The status of the SDS pool can change while sdss is running; the picture it gives is only a close approximation to reality.

SEE ALSO

privtext(1), ps(1)

General UNICOS System Administration, Cray Research publication SG–2301

NAME

sed – Invokes the stream editor

SYNOPSIS

sed [-g] [-n] *script* [*files*]
sed [-g] [-n] [-e *script*]... [-f *sfile*]... [*files*]

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4
AT&T extension (-g option)

DESCRIPTION

The sed utility copies *files* (standard input by default) to standard output, edited according to a script of commands. The script is obtained from either the *script* operand string or a combination of the arguments from the -e *script* and -f *sfile* options.

The sed utility accepts the following options:

-e *script* Adds the editing commands specified by the *script* argument to the end of the script of editing commands. The *script* argument has the same properties as the *script* operand.

-f *sfile* Adds the editing commands from *sfile* to the end of the script.

-g For every substitute (s) command, performs a global replacement.

-n Suppresses the default output. Only lines explicitly selected for output are written.

files Files to be copied.

The -f option causes the script to be taken from file *sfile*; these options accumulate. If only one -e option and no -f options are specified, you can omit the -e flag.

A script consists of editing commands, one per line, of the following form:

[*address* [, *address*]] *function* [*arguments*]

In normal operation, sed cyclically copies a line of input, less its terminating <newline>, into pattern space (unless something is left after a D command), applies in sequence all commands whose addresses select pattern space, and at the end of the script, copies the pattern space to the standard output (except under -n) and deletes the pattern space.

Some of the commands use hold space to save all or part of pattern space for subsequent retrieval.

address is a decimal number that counts input lines cumulatively across files, a `$` that addresses the last line of input, or a context address, that is, a */ regular\xpression /* in the style of `ed`(1) and modified as follows:

- In a context address, the construction `\?`*regular expression*`?`, in which *?* is any character, is identical to */ regular expression /*. In the context address `\xabc\xdefx`, the second `x` stands for itself; therefore, the regular expression is `abcxdef`.
- Escape sequence `\n` matches a `<newline>` character embedded in the pattern space.
- A period `.` matches any character except the terminal `<newline>` character of the pattern space.
- A command line with no addresses selects every pattern space.
- A command line with one address selects each pattern space that matches the address.
- A command line that has two addresses selects the inclusive range from the first pattern space that matches the first address through the next pattern space that matches the second. (If the second address is a number less than or equal to the line number first selected, only one line is selected.) Thereafter, the process is repeated, looking again for the first address.

Editing commands can be applied only to nonselected pattern spaces by using the negation function `!` described in this section.

In the following list of functions, parentheses enclose the maximum number of permissible addresses for each function.

The *text* argument consists of one or more lines, all but the last of which end with `\` to hide the `<newline>`. Backslashes in text are treated like backslashes in the replacement string of an `s` command, and may be used to protect initial `<blank>`s and `<tab>`s against the stripping that is done on every script line. The *rfile* or *wfile* argument must terminate the command line and must be preceded by one or more `<blank>`s. Each *wfile* is created before processing begins. There can be at most 10 distinct *wfile* arguments.

(1)`a\`
text Appends. Place *text* on the output before reading the next input line.

(2)`b` *label* Branches to the `:` command bearing the *label*. If *label* is empty, branch to the end of the script.

(2)`c\`
text Changes. Deletes the pattern space. With zero or one address or at the end of a two-address range, place *text* on the output. Start the next cycle.

(2)`d` Deletes the pattern space. Starts the next cycle.

(2)`D` Deletes the initial segment of the pattern space through the first `<newline>`. Starts the next cycle.

(2)`g` Replaces the contents of the pattern space by the contents of the hold space.

(2)`G` Appends the contents of the hold space to the pattern space.

(2)h Replaces the contents of the hold space by the contents of the pattern space.

(2)H Appends the contents of the pattern space to the hold space.

(1)i\
text Insert. Place *text* on the standard output.

(2)l Lists the pattern space on the standard output in an unambiguous form. Nonprinting characters are written in 2-digit ASCII, and long lines are folded. (Standard escapes (see the following table) are used for things such as <tab> and <form-feed>.)

007	\a	<alert>
010	\b	backspace
011	\t	<tab>
012	\n	<newline>†
013	\v	<vertical-tab>
014	\f	<form-feed>
015	\r	<carriage-return>

† The sed utility cannot produce this.

(2)n Copies the pattern space to the standard output. Replace the pattern space with the next line of input.

(2)N Appends the next line of input to the pattern space with an embedded <newline>. (The current line number changes.)

(2)p Print. Copies the pattern space to the standard output.

(2)P Copies the initial segment of the pattern space through the first <newline> to the standard output.

(1)q Quit. Branches to the end of the script. Do not start a new cycle.

(2)r *rfile* Reads the contents of *rfile*. Place them on the output before reading the next input line.

(2)s/*regular expression*/*replacement*/*flags*
Substitutes the *replacement* string for instances of the *regular expression* in the pattern space. You can use any character other than backslash or <newline> instead of /. For a complete description, see ed(1). *flags* is zero or more of the following:

n *n*=1-512. Substitutes for just the *n*-th occurrence of the *regular expression.*

g Global. Substitutes for all nonoverlapping instances of the *regular expression* rather than just the first one.

p Prints the pattern space if a replacement was made.

w *wfile* Write. Appends the pattern space to *wfile* if a replacement was made.

(2)t *label* Test. Branches to the : command that bears *label* if any substitutions were made since the most recent reading of an input line or execution of a t. If *label* is empty, branch to the end of the script.

(2)`w` *wfile* Write. Appends the pattern space to *wfile*.

(2)`x` Exchanges the contents of the pattern and hold spaces.

(2)`y`/*string1*/*string2*/
Transform. Replaces all occurrences of characters in *string1* with the corresponding character in *string2*. The lengths of *string1* and *string2* must be equal.

(2)`!` *function* Negation. Applies *function* (or group, if *function* is `{`) only to lines not selected by the address(es).

(0)`:` *label* This command does nothing; it bears a *label* to which `b` and `t` commands can branch.

(1)`=` Places the current line number on the standard output as a line.

(2)`{` Executes the following `sed` commands through a matching `}` only when the pattern space is selected. The list of `sed` commands are separated by `<newline>`s. The `{` can be preceded with `<blank>`s and can be followed with white space. The *commands* may be preceded with white space. The terminating `}` must be preceded by a `<newline>` and then zero or more `<blank>`s.

(0) An empty command is ignored.

(0)# If # appears as the first character on the first line of a script file, that entire line will be treated as a comment, except if the character after the # is `n`, in which case the default output is suppressed. The rest of the line after #`n` is also ignored. A script file must contain at least one noncomment line.

NOTES

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
`system`, `secadm`	In a privileged administrator shell environment, shell-redirected I/O is not subject to file protections.
`sysadm`	Shell-redirected output is subject to security label restrictions.

If the `PRIV_SU` configuration option is enabled, shell-redirected I/O on behalf of the super user is not subject to file protections.

EXIT STATUS

The `sed` utility exits with one of the following values:

0 Successful completion.

>0 An error occurred.

SEE ALSO

awk(1), ed(1), grep(1)

regex(3C) in the *UNICOS System Libraries Reference Manual*, Cray Research publication SR–2080

sed & awk, Dale Dougherty, O'Reilly & Associates, Inc., 1990.

The UNIX Programming Environment, Brian W. Kernighan and Rob Pike, Prentice-Hall, Inc., 1984

NAME

segldr – Invokes the Cray Research segment loader (SEGLDR)

SYNOPSIS

segldr [-A *file*] [-a] [-b *value*] [-D *dirstring*] [-E] [-e *name*] [-F] [-f *value*] [-g] [-H *hi*[+*he*]] [-i *dirfiles*] [-j *names*] [-k] [-L *ldirs*] [-l *names*] [-M *arguments*] [-m] [-N] [-n] [-O *keyword*] [-o *outfile*] [-S *si*[+*se*]] [-s] [-t] [-u *unames*] [-V] [-z *file*] [-Z] *files*

IMPLEMENTATION

Cray PVP systems

DESCRIPTION

The segldr utility links relocatable object modules to produce an executable program. Load maps, if selected, are written to the stdout file by default (see the -M option). Error messages are written to the stderr file by default (see the -k option). The output file is execute-enabled when no warning or fatal errors occur during the load.

The segldr utility accepts the following options:

-A *file* Specifies executable file containing symbol information for SEGLDR. Symbol references in newly loaded code are linked to addresses in the existing executable code. The resultant output file then contains a code fragment that can execute in the existing program address space.

-a Aligns all code and local data blocks on instruction buffer boundaries.

-b *value* Adds 1024 times *value* number of words to the BSS (uninitialized data) area of the loaded program.

-D *dirstring* Passes *dirstring* of SEGLDR directives to SEGLDR. The *dirstring* argument is a character string of global SEGLDR directives separated with semicolons.

-E Echoes all processed directives to the load map file. See the -M option.

-e *ename* Sets the program entry address to the value of symbol *ename*.

-F Loads all modules from bin files, whether or not they are referenced.

-f *value* Fills uninitialized, statically allocated areas of the program with *value*. The *value* argument may be one of the following:

- zeros Fills with 0 bits (default).
- ones Fills with 1 bits.
- indef Fills with 0605054000000000000000 octal, to cause a floating-point error if referenced.

`-indef` Fills with 1605054000000000000000 octal, to cause a floating-point error if referenced.

`indefa` Sets uninitialized data to the product of a logical OR operation of O'0605054000000000000000 multiplied by the address of the word being preset. This value is the same as that of `indef`, except that the address of the word referenced will appear in the low-order bits of the value.

`-indefa` Sets uninitialized data to the product of a logical OR operation of O'1605054000000000000000 multiplied by the address of the word being preset. This value is the same as that of `-indef`, except that the address of the word referenced will appear in the low-order bits of the value.

A 16-bit octal value
Stores the value in each parcel of each uninitialized word.

`-g` Generates the Debug Symbol tables and appends them to the executable file. This option is enabled by default (see the `-s` option).

`-H` *hi*[+*he*] Assigns initial heap size (*hi*) and heap expansion increment (*he*). Specify sizes in words.

`-i` *dirfiles* Reads and processes the directives in the directives files. *dirfiles* contains a list of directives file names, separated by commas. When a `-` is present as one of the file names, the `segldr` utility reads the `stdin` file for directives. When a name begins with a `.` or `/` symbol, the loader assumes it is a complete path name and uses it without modification. Otherwise, the loader checks for the named files in the current directory.

`-j` *names* Reads and processes the directives in the directives files. *names* contains a list of directives file names, separated by commas. When a name begins with a `.` or `/` symbol, the loader assumes it is a complete path name and uses it without modification. Otherwise, the loader checks for a `segdir/`*name* file in the list of search directories and uses the first one found. See the `-L` option for the list of search directories.

`-k` Redirects all but summary-class error messages to the load map file. See the `-M` option.

`-L` *ldirs* Changes the `-l` option search algorithm to look for library files in directories *ldirs* before looking in the `/opt/ctl/craylibs/craylibs`, `/lib`, or `/usr/lib` directories. If the `-F` option is used to include the system default directories, the loader searches directories *ldirs* for those libraries before searching the `/opt/ctl/craylibs/craylibs`, `/lib`, or `/usr/lib` directories. Multiple `-L` options are cumulative.

`-l` *names* Identifies library files. When a name begins with a `.` or `/` symbol, it is assumed to be a full path name, and the `segldr` utility uses it as is. Otherwise, the `segldr` utility checks first for files `/opt/ctl/craylibs/craylibs/lib`*name*`.a` and `/lib/lib`*name*`.a`, and then for file `/usr/lib/lib`*name*`.a`. It uses the first one found. See the `-L` option.

`-M` *file* or `-M`, *opts* or `-M` *file,opts*
Selects an optional load map file, *file*, and the type of map to produce. If *file* is present, the `segldr` utility writes the load maps to that file in a paginated format, 132 characters per line. If a file is not provided, the `segldr` utility writes the load maps to the `stdout` file in a nonpaginated format, 80 characters per line. Load map options (*opts*) are as follows:

`s` or `stat`	Lists only load statistics
`a` or `address`	Sorts block map by address (the default map, if no *opt* is specified)
`al` or `alpha`	Sorts block map by name
`b` or `brief`	Restricts maps to `bin` files only
`c` or `cbxrf`	Lists common-block cross-references
`e` or `epxrf`	Lists entry-point cross-references
`p` or `part`	Lists a combination of `address` and `alpha`
`f` or `full`	Lists all load maps

`-m` Generates the address-level load map and writes it to the `stdout` file. This option is equivalent to the `-M`, `address` option.

`-N` Inhibits the inclusion of the default libraries in the load.

`-n` Generates a shared-text program.

`-O` *keyword* Selects allocation order. The *keyword* variable can be the following:

`tdb`	Allocates all code, followed by all initialized data, followed by all uninitialized data

`-o` *outfile* Writes executable program to the *outfile* file. When the `-o` option is not used, the executable program is written to the file named by the `ABS` directive. When neither the `-o` option nor `ABS` is specified, the executable output is written to file `a.out`.

`-S` *si*[+*se*] Assigns initial stack size (*si*) and stack expansion increment (*se*). Specify sizes in words.

`-s` Inhibits the generation of Debug Symbol tables. These are typically generated by default.

`-t` Executes in trial mode. The `segldr` utility scans all object modules and generates load maps, but it does not produce an executable program.

`-u` *unames* Enters *unames* as undefined symbols. This is useful for loading from a library, because undefined symbols are needed to force the loading of desired routines.

`-V` Lists SEGLDR's version line to the `stderr` file.

`-z` *file* Specifies an alternative default directives file. The alternative directives must configure the program correctly for execution under the UNICOS operating system.

`-Z` Inhibits the loader from reading the default directives file, either `/lib/segdirs/def_seg` or `/opt/ctl/craylibs/craylibs/segdirs/opt_defseg`. The default directives file is required for configuring programs correctly for execution under the UNICOS operating system. The `-Z` option should be used only by special-purpose programs.

files Specifies files to be loaded. These files can contain sequential object modules produced by the compilers or the assembler, or they can be object module files prepared by the `ar`(1) command or the `bld`(1) utility. Naming files on the command line has the same effect as naming them in a `BIN` directive. Files ending with `.o` will be treated as `bin` files. Files ending with `.a` will be treated as `lib` files. For compatibility, files containing directives may also be specified. The `-i` option is recommended for that purpose. It is also recommended that you create `bld` library archives rather than `ar` archives for use with Cray Research loaders.

`bin` and `lib` Files

You can direct the `segldr` utility to process an object file as either a `bin` or a `lib` file. You can specify `bin` files as arguments on the command line or by using the `BIN` directive. Name `lib` files with the `-l` option on the command line or with the `LIB` directive. The `segldr` utility processes both types of files in essentially the same manner. The `segldr` utility scans all object files and notes calling relationships. Beginning at the main program in the calling tree, `segldr` retains all modules required by the program and discards all others. Differences between `bin` and `lib` processing primarily involve the following items:

- Processing order (all `bin` files are processed before all `lib` files)
- The `-F` option or `FORCE` directive (does not affect `lib` files)
- The `-M` *file*, `brief` option, or `MAP=BRIEF` directive (lists load modules derived only from `bin` files)
- Fortran `BLOCK DATA` subprograms (always included from `bin` files but only if reference from `lib` files)
- The `DUPENTRY` directive (message level for three cases: both in `bin` files, `bin` file and `lib` file, both in `lib` files)
- The `DUPORDER` directive (if in both `bin` files and `lib` files, chooses entry point from `bin` files)
- C programs containing initialized global data (always included from `bin` files but only if referenced from `lib` files)

Both `bin` and `lib` files can contain either sequential object modules created by the compilers and the assembler, or libraries prepared by `bld`(1) and `ar`(1).

Default System Library Files

After processing all object files and any libraries supplied by the user, the `segldr` utility scans the default system library files, unless inhibited by the `-N` option or `NODEFLIB` directive. The following list shows the default library files for all Cray Research systems. The list shows the libraries in the order in which the loader searches them:

`libc.a` C library
`libu.a` Utilities library

`libm.a`	Math library
`libf.a`	Fortran library
`libfi.a`	Fortran intrinsic library
`libsci.a`	Science library
`libp.a`	Pascal library

Some of the default libraries listed may be released separately from the UNICOS operating system; therefore, they may not be present on your system. Missing libraries are silently ignored.

If you do not specify the `-L` option, the `segldr` utility looks for the default library files first in the `opt/ctl/craylibs/craylibs` and `/lib` directories, then in `/usr/lib`. If you have used one or more `-L` options, the `segldr` utility looks in all directories specified by these options, and then looks in `/lib`, `opt/ctl/craylibs/craylibs`, and `/usr/lib`. You can use the `defdir` directive to change the default directories.

The `segldr` utility usually reads directives from files supplied as option-arguments to the `-i` option. For compatibility, however, if a file provided as a command-line argument does not end in `.o` or `.a`, the `segldr` utility checks its contents. If the file contains ASCII characters, the `segldr` utility will process it as a directives file; otherwise, `segldr` processes it as an object file.

ENVIRONMENT VARIABLES

The `segldr` utility looks for and processes the following environment variables:

`SEGDIR`	Contains one or more strings separated by semicolons. Each string may be either a `segldr` directive or the name of a file containing `segldr` directives.
`TMPDIR`	Specifies the directory that the loader uses for its temporary file. The default directory may be specific to each system.
`LPP`	Specifies the number of lines to print on each page of listing output. The value must be between 15 and 999, and the default is 57.
`MSG_FORMAT`	Describes a format specification similar to that of C library routine `printf`; this specification can be used to alter `segldr` error message displays.
`NLSPATH`	Specifies a list of alternative directories that the loader should search for its error message catalog. It is used to select alternative catalogs for debugging, or when different versions of `segldr` are operating on the same system. `NLSPATH` is not needed for normal operations.
`TARGET`	Specifies the machine characteristics of the system on which the program will execute. If the `TARGET` variable has not been specified, the program will be adapted to the host system.

segldr Directives

The following is a summary of `segldr` directive syntax rules. For more complete descriptions of the directives and their use, see the *Segment Loader (SEGLDR) and ld Reference Manual*, Cray Research publication SR–0066.

- *keyword=value*
- Directives can be uppercase or lowercase but not mixed case.

- Comments can appear anywhere (an asterisk (*) indicates the beginning of a comment).
- Directives are terminated by semicolon(;), asterisk (*), or end-of-line character.
- More than one directive separated by a semicolon (;) can appear on a line.
- Directives cannot be longer than 256 characters.
- Elements in a list must be separated with commas.
- Null directives are ignored.

The following is a list of all segldr directives available under the UNICOS operating system and a brief description of each. For more complete descriptions of the directives and their use, see the *Segment Loader (SEGLDR) and ld Reference Manual*, Cray Research publication SR–0066.

abs	Specifies the file to receive the executable program.
addbss	Expands the initial size of the program.
align	Controls the starting locations of modules and common blocks.
bin	Names relocatable object input files to be searched.
calltree	Defines the start of the block of calling-tree definition directives (see endct).
callxfer	Names the external symbol used by the system startup routine to call the xfer entry.
case	Determines whether characters in the directives file are converted to uppercase before they are processed.
comment	Annotates segldr directives (* character).
commons	Loads the listed common blocks in the specified order (see scommons).
compress	Sets the threshold for compression of executable files.
copy	Forces a segmented program to execute from a scratch file.
cpucheck	Determines whether segldr performs machine-characteristic checking.
defdir	Specifies default directory search lists.
defheap	Specifies the minimum heap size and heap increment value for all programs.
deflib	Specifies extra libraries for segldr to search in addition to the default system libraries.
defstack	Sets the default program stack size.
dup	Lets segldr load modules of the same name into different segments.
dupentry	Specifies the severity level of messages for duplicated-entry point errors.
dupload	Specifies the severity level of messages for common-block initialization by more than one module.
duporder	Selects the method segldr uses to process duplicated entry points found in libraries.
dynamics	Names the common block that can expand or contract under user control.

`echo`	Resumes or suppresses the display of input directives.
`endct`	Defines the end of the block of calling-tree definition directives (see `calltree`).
`endseg`	Terminates a segment description (see `segment`).
`endtree`	Terminates the set of segment tree definition directives.
`equiv`	Substitutes a call to one entry point for a call to another.
`float`	Specifies the movable block-positioning algorithm for segmented programs.
`force`	Forces modules to be loaded, even if they are not called in execution.
`freeheap`	Specifies the minimum amount of free memory available in the heap after the initial stack allocation.
`hardref`	Converts all soft references to hard references for specified symbols.
`heap`	Allocates memory that the heap manager can manage dynamically.
`hidesym`	Specifies a global symbol that is not to be visible.
`incfile`	Identifies a symbol input file.
`include`	Identifies a directives file to be included.
`keepsym`	Specifies a global symbol that is to be visible; all other symbols are not visible.
`lbin`	Specifies relocatable object input files to be searched.
`lib`	Names the files for `segldr` to search when looking for entry points referenced in `bin` files.
`libdir`	Specifies directories other than the default to search for system libraries.
`linclude`	Specifies a file that should be included in the load process.
`llib`	Names the files for `segldr` to search when looking for entry points referenced in `bin` files using only the file name component.
`logfile`	Identifies the log messages file.
`loguse`	Identifies object files or libraries that should be logged.
`map`	Specifies the load maps to be generated by `segldr`.
`mlevel`	Specifies the severity level of messages in the listing output.
`modules`	Names the modules to be loaded (see `smodules`).
`msglevel`	Selects message severity level for specific messages.
`nodeflib`	Ignores all default libraries when loading.
`nodupmsg`	Suppresses duplicate symbol messages for specific symbols.
`nousxmsg`	Suppresses unsatisfied symbol messages for specific symbols.
`omit`	Identifies modules that should be excluded from the program.

`order` Lets you determine the central memory allocation method `segldr` uses.

`org` Sets the initial address for different portions of the program.

`outform` Specifies the type of the output file.

`preset` Specifies a value used to preset uninitialized data areas.

`redef` Specifies the severity level of messages for redefined-common-block errors.

`save` Specifies whether the current segment states for segments are written to mass storage before `segldr` overlays them with other segments.

`scanner=` `ON` | `OFF`
Scans a program targeted for a CRAY EL98 or CRAY J90 system and detects and corrects potential problems. The default on CRAY EL98 and CRAY J90 systems is `ON`.

`scanpad=` *nnnnnn*
Adds additional unused memory to a program being scanned for potential problems (see the `scanner` *directive).*

`scommons` Loads the listed common blocks in the specified order (no error messages issued).

`segment` Names the segment being described by the segment description directives (see `endseg`).

`segorder` Lets you determine the order of the segments in the executable file.

`set` Assigns a value to an entry point. You can use the `set` directive to define the library buffer size for different file structures. For more information, see the

`slt` Specifies the size of the Segment Linkage table (SLT).

`smodules` Names the modules to be loaded (no error messages issued).

`softref` Converts all hard references to soft references for specific symbols.

`stack` Sets the program stack size.

`start` Names the entry point at which the program begins executing.

`symbols` Determines whether a Debug Symbol table is generated.

`system` Selects the target operating system on which the program will be run.

`title` Specifies a page header for load maps.

`tree` Begins the set of segment tree definition directives.

`trial` Makes a sample `segldr` run without creating an executable program.

`tstack` Sets the slave task stack size for a multitasked program.

`unsat` Names unsatisfied references to be loaded from libraries.

`usx` Specifies the severity level of messages for unsatisfied-external-symbol errors.

`xfer` Names the user program entry point to which the system startup routine transfers control.

`zerocom` Specifies the name of the common block to be placed at the zero address of the data space.

`zerodata` Specifies the name of the module to be placed at the zero address of the local data space.

`zerotext` Specifies the name of the module to be placed at the zero address of the text space.

`zsyms` Specifies whether or not the loader is to include the `zzzzzz??` symbols in the load module.

MESSAGES

The full range of `segldr` error messages and the proper responses to them are listed in the *Segment Loader (SEGLDR) and ld Reference Manual*, Cray Research publication SR–0066.

FILES

`a.out`	Executable program
file`.o`	Relocatable object file
`/opt/ctl/craylibs/craylibs/libf.a`	Fortran library
`/opt/ctl/craylibs/craylibs/libfi.a`	Fortran intrinsic library
`/opt/ctl/craylibs/craylibs/libm.a`	Math library
`/opt/ctl/craylibs/craylibs/libsci.a`	Scientific library
`/lib/libc.a`	C library
`/opt/ctl/CC/CC/lib/libC.a`	C++ library (only if your site has a C++ license)
`/lib/libp.a`	Pascal library
`/opt/ctl/craylibs/craylibs/libu.a`	Utility library
`/lib/segdirs/def_seg` and `/opt/ctl/craylibs/craylibs/segdirs/opt_def_seg`	Default directives files

SEE ALSO

`ar`(1) archive and library maintainer for portable archives
`bld`(1) maintains relocatable libraries
`cc`(1) invokes the Cray Standard C compiler
`ld`(1) invokes the link editor with traditional UNIX invocation
`nm`(1) prints name list from load modules
`pascal`(1)invokes the Pascal compiler

`f90`(1) invokes the CF90 compiler

`mppld`(1) invokes the Cray Research MPP loader with traditional UNIX invocation
`mppldr`(1) invokes the Cray Research MPP loader

`a.out`(5) describes the loader output file
`mpp.a.out`(5) describes the MPP loader output file
`relo`(5) describes the relocatable object table format under the UNICOS operating system
`taskcom`(5) describes the task common table format
in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

Segment Loader (SEGLDR) and ld Reference Manual, Cray Research publication SR–0066

NAME

`setf` – Initializes a file

SYNOPSIS

`setf` [`-c`] [`-n` *size*[:*units*]] [`-p` *parts*] *file*

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `setf` utility initializes a new or existing file. The file is created if it does not already exist, and the specified number of bytes or blocks are allocated to it. By option, the user may specify that the blocks allocated must be contiguous and on which partition of the file system allocation is to be attempted.

The `setf` utility does not make changes to the inode unless preallocation is requested.

The `setf` utility accepts the following options:

`-c` Forces the program to fail if blocks cannot be allocated contiguously.

`-n` *size*[:*units*]
Indicates the total number of bytes or, if followed by the letter `b`, the total number of blocks to be allocated. The optional :*units* subfield indicates the minimum number of bytes or, if followed by the letter `b`, the minimum number of blocks to allocate per partition requested. (See the `-p` option.)

`-p` *parts* Indicates the partitions of the file system on which allocation is to be attempted. There is no guarantee that blocks will actually be allocated on the specified partitions. The *parts* field may be entered as a single number, a range (*m*–*n*), a set (*m*:*n*), or a combination of ranges and sets (see example). The dash (–) in the range specifies a range of partitions to be used (for example, 2–5 means partitions 2 through 5). A colon (:) in the set specifies a list of partitions to be used (for example, 2:4:6 means partitions 2, 4, and 6).

The partition numbers are submitted directly through `ialloc`(2) system calls. This option achieves striping of the file on the specified partitions.

files Specifies the name of the file to call or create.

NOTES

If the :*units* subfield is used on the `-n` option, and there are fewer partitions specified than will satisfy the total size if applied, (product of :*units* * *parts* is less than *size*), `setf` will repeat through the partition list in a circular fashion until the specified total size of the file is allocated. This technique is called *striding*.

If the -p option is used, and the :*units* subfield was not used on the -n option, setf will attempt to allocate the *size* equally across the total number of partitions specified by -p.

The setf command uses ialloc(2), which allocates space in multiples of allocation unit size for the file system.

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
system, secadm	Allowed to initialize any file. In a privileged administrator shell environment, shell-redirected I/O is not subject to file protections.
sysadm	Allowed to initialize any file subject to security label restrictions. Shell-redirected I/O is subject to security label restrictions.

If the PRIV_SU configuration option is enabled, the super user is allowed to initialize any file. Shell-redirected I/O on behalf of the super user is not subject to file protections.

EXAMPLES

The following setf command attempts to allocate 16,000 blocks (in units of 1000 blocks) on partitions 0 through 4, 6, and 8 through 17.

```
setf -n 16000b:1000b -p 0-4:6:8-17 myfile
```

SEE ALSO

assign(1), df(1), fck(1)

ialloc(2), open(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

NAME

`setucat` – Sets your active categories

SYNOPSIS

`setucat` *cats*

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `setucat` utility activates one or more of your authorized categories. Your active category identifies the administrative role under which you are currently functioning. Your active categories are a subset of your authorized categories. Your authorized categories are initialized in the user database (UDB) by an appropriately authorized administrator.

The `setucat` utility accepts the following argument:

cats Specifies the category to be activated.

The *cats* argument consists of one of the following elements:

- Category name. The category name identifies a category to be made active.
- Comma-separated list of category names.
- A category bit mask (octal); the category bit mask is the bit value corresponding to one or more categories to be activated. This argument must be expressed as an octal number.
- The name `none`, which sets your active category to 0, is also valid. You may set your active category to 0.

The `setucat` utility can fail for one or more of the following reasons:

- The requested category is not valid.
- The requested category is not a subset of your authorized categories.

NOTES

All `setucat` requests are recorded in the security log, along with an indication of success or failure.

EXAMPLES

Example 1: In the following example, the name `syscat1` represents the category that has an octal bit mask of 040.

```
setucat 040
```

Example 2: The following examples set the `syscat1` category as your active category:

```
setucat syscat1
```

Example 3: The following example sets the `secadm` and `sysfil` categories:

```
setucat secadm,sysfil
```

Example 4: The following example deactivates an active category for a user:

```
setucat 0
```

SEE ALSO

`sh`(1), `spset`(1)

`setucat`(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

`slog`(4), `slrec`(5) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

General UNICOS System Administration, Cray Research publication SG–2301

NAME

setucmp – Sets your active compartments

SYNOPSIS

setucmp *cmps*

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The setucmp utility adds compartments to your active compartment set. Your active compartments determine, in part, your current file access capability. Your active compartments are a subset of your authorized compartments. Your authorized compartments are initialized in the user database (UDB) and the network access list (NAL) by an appropriately authorized administrator.

The setucmp utility accepts the following argument:

cmps Specifies compartments to be activated.

The *cmps* argument consists of one of the following elements:

- A list of one or more compartment names
- A compartment bit mask (octal)

Each compartment name identifies a compartment to be made active. Multiple compartment names must be separated by a comma (no spaces).

The compartment bit mask is the union of bit values corresponding to each compartment to be activated. This argument must be expressed as an octal number.

The *cmps* argument may also consist of the word ALL, which activates all of your authorized compartments.

The setucmp utility fails for the following error conditions:

- The requested compartments are not authorized for use on the UNICOS system.
- The requested compartments are not a subset of your authorized compartments.
- Activating the requested compartments will create an access violation with existing open files (character special files owned by the user are a special case).
- The request is not issued from the login shell process.
- There are no other processes running in the background (the process must be the master process).

NOTES

You cannot deactivate a compartment once it has been activated. In a privileged shell environment, users with an active `system` or `secadm` category are allowed to set their active compartments to any defined value. If `PRIV_SU` is enabled, the super user is allowed set its active compartments to any defined value.

The compartments of open character special files (ttys) owned by the user are automatically set to the new active compartments.

All successful requests to set your compartments are recorded in the security log and, if mandatory access violation logging is enabled, all unsuccessful requests are recorded in the security log.

EXAMPLES

In the following examples, the name `classified` represents the compartment with an octal bit mask of 020. Also, the name `confidential` represents the compartment with an octal bit mask of 04.

Example 1: The following examples add the `confidential` compartment to your active compartments:

```
setucmp confidential
setucmp 04
```

Example 2: The following examples add the `confidential` and `classified` compartments to your active compartments:

```
setucmp confidential,classified
setucmp 024
```

Example 3: The following example activates all of your authorized compartments:

```
setucmp ALL
```

SEE ALSO

`setulvl`(1), `sh`(1), `spset`(1)

`setucmp`(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

`slog`(4), `slrec`(5) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

General UNICOS System Administration, Cray Research publication SG–2301

NAME

`setulvl` – Raises your active security level

SYNOPSIS

`setulvl` *level*

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `setulvl` utility raises your active security level. Your *active security level* is the security level at which you are currently operating, and determines, in part, your file access capability.

Your active security level is set within your authorized minimum and maximum security level range. Your authorized security level range is initialized in the user database (UDB) and the network access list (NAL) by an appropriately authorized administrator.

The `setulvl` utility accepts the following argument:

level Specifies the security level to be activated. *level* can be a number from 0 through 16, where 16 is the highest security level allowed. It can also be the name of a security level; level names are established by an appropriately authorized administrator. The requested security level must not be less than your current active security level.

The `setulvl` utility can fail for one or more of the following reasons:

- The requested level is not authorized for use on the UNICOS system.
- The requested level does not fall within the your authorized minimum and maximum security level range.
- The requested level is less than your active security level.
- The requested level creates an access violation with existing open files (character special files owned by a user are a special case).
- The request is not issued from the login shell process.
- There are other processes running in the background (for `setulvl` to execute correctly, the only process that can be running is the login shell process).

To validate your request, `setulvl` checks the minimum and maximum security levels assigned to you at login against the system's lower and upper security levels.

NOTES

You can only raise your active security level. In a privileged shell environment, users with an active `system` or `secadm` category are allowed to set their security level to any defined value. If `PRIV_SU` is enabled, the super user is allowed set its security level to any defined value.

EXAMPLES

In the following example, the name `classified` represents security level 3. The commands shown will raise your active security level to level 3:

```
setulvl classified
```

or

```
setulvl 3
```

SEE ALSO

`setucmp`(1), `sh`(1), `spset`(1)

`setulvl`(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

`udb`(5) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

General UNICOS System Administration, Cray Research publication SG–2301

NAME

setusrv – Sets your authorized security attributes

SYNOPSIS

setusrv [-c *valcmp*] [-i *maxcls*] [-j *valcat*] [-l *minlvl*] [-p *permit*] [-u *maxlvl*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The setusrv utility sets your authorized security attributes (that is, security levels, security compartments, integrity classes, categories, and permissions).

These authorized security attributes provide the range within which you may work. The authorized security attributes are initially determined by the security administrator.

The setusrv utility accepts the following options and arguments:

-c *valcmp* Sets your authorized compartments to *valcmp*. *valcmp* must be a comma-separated list of compartment names (no spaces) or an octal mask where each bit represents a compartment to be authorized.

-i *maxcls* Sets your maximum integrity class to *maxcls*. This option is not supported.

-j *valcat* Sets your authorized categories to *valcat*. *valcat* must be a comma-separated list of category names (no spaces) or an octal bit mask where each bit represents a category to be authorized.

-l *minlvl* Sets your minimum security level to *minlvl*. *minlvl* must be a security level number or name.

-p *permit* Sets your permissions to *permit*. *permit* must be a comma-separated list of permission names (no spaces) or an octal bit mask where each bit represents a permission to be set.

-u *maxlvl* Sets your maximum security level to *maxlvl*. *maxlvl* must be a security level number or name.

The setusrv utility can fail for one or more of the following reasons:

- An attempt is made to expand your minimum or maximum security level range.
- An attempt is made to expand your maximum integrity class.
- An attempt is made to expand your authorized compartment set.
- An attempt is made to expand your authorized category set.
- The requested maximum security level is less than the requested minimum security level.
- The requested minimum or maximum security level range is out of range of the UNICOS system minimum and maximum level range.
- The requested maximum integrity class is less than 0.

- The requested authorized compartments are out of range for the UNICOS system's set of authorized compartments.

NOTES

If the requested minimum or maximum security level or integrity class values are outside those authorized for the UNICOS system, they are silently brought within the bounds of the system.

If the requested authorized compartments, categories, or permissions are outside those authorized for the UNICOS system, they are silently brought within the bounds of the system.

All `setusrv` requests are recorded in the security log, along with an indication of success or failure.

EXAMPLES

Example 1: In the following examples, the names `classified` and `secret` represent the security levels 3 and 7, respectively. The following examples constrict your minimum or maximum security level range to levels 3 through 7:

```
setusrv -l classified -u secret
```

or

```
setusrv -l 3 -u secret
```

or

```
setusrv -l 3 -u 7
```

Example 2: In the following examples, the names `red` and `green` represent the compartments whose octal bit masks are 01 and 0400, respectively. The following examples constrict your authorized compartments to `red` and `green`:

```
setusrv -c green,red
```

or

```
setusrv -c 0401
```

Example 3: In the following examples, the names `special` and `priority` represent the categories whose octal bit masks are 020 and 0100, respectively. The following examples constrict your authorized categories to `special` or `priority`:

```
setusrv -j priority,special
```

or

```
setusrv -j 0120
```

Example 4: In the following examples, the name `suidgid` represent the permission whose octal bit masks are 0100. The following examples constrict your permissions to `suidgid`:

```
setusrv -p suidgid
```

or

```
setusrv -p 0100
```

Example 5: The following command line performs the operations illustrated by all of the preceding examples:

```
setusrv -l 3 -u 7 -c red,green -j special,priority -p 0100
```

SEE ALSO

`setucat`(1), `setucmp`(1), `setulvl`(1), `sh`(1), `spset`(1)

`setucat`(2), `setucmp`(2), `setulvl`(2), `setusrv`(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

`slog`(4), `slrec`(5) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

General UNICOS System Administration, Cray Research publication SG–2301

NAME

shrview – Displays detailed fair-share scheduler information

SYNOPSIS

shrview [-c] [-d *type*] [-o *sort*] [-r *rate*] [-s *crit*] [*ID*] [*ID*] ... [id] ...

IMPLEMENTATION

Cray PVP systems

DESCRIPTION

The shrview utility is an integrated tool for displaying information about the behavior and current state of the fair-share scheduler (also referred to as *fair-share*).

Many different display options and formats are available. Lnode information is available in several different formats, each designed to illustrate specific aspects of the fair-share scheduler. The order and selection of the lnode information can be configured. Additional displays are available to list fair-share parameters, statistics, and internal tables.

Selection and configuration of displays can be done interactively when in curses mode (see the curses(3) man page).

The shrview utility accepts the following options:

-c Enables continuous curses display.

-d *type* Selects display type. The following display types are available:

- a (ADJGROUP) Display tailored to show the effect of the ADJGROUPS flag and mingshare parameter on individual users or lnodes.
- b (brief) Abbreviated summary of user or lnode information that will fit 2 columns per 80-column page.
- c (shconsts) Dump of all fields in the kernel shconsts structure.
- l (LIMSHARE) Display tailored to show the effect of the LIMSHARE flag and maxushare parameter on individual users or lnodes.
- m (Monitor) Display of internal lnode fields related to housekeeping, which are not displayed in other displays.
- n (Nice tables) List of values in internal fair-share tables for each nice value.
- p (Params) Display of current setting of all fair-share parameters that can be set with the shradmin(8) command.
- r (Rates) Graphical representation of user or lnode information. A horizontal bar represents the relationship of current priority, user share, and current processing rate. Priority is represented with a p, user share with an s, and processing rate with a bar of # characters.

s (Statistics) Display of fair-share statistics.

v (View) Display of user or lnode information. This is the default display type. Useful indication of individual users' current fair-share priority and contributing factors.

w (Wide) A wide (132-column) display consisting of information from the view (`-dv`), `ADJGROUPS` (`-da`), monitor (`-dm`), and `LIMSHARE` (`-dl`) displays.

`-o` *sort* Selects lnode sort option. Both groups and user or account lnodes are sorted, but members of the same group will always be listed together. The following sort options are available:

c (Charge) Sorts lnodes by cumulative charges.

i (ID) Sorts lnode display based on alphabetical order of lnode names. This is the default.

p (Priority) Sorts lnodes by relative priority.

s (Share allocation) Sorts lnodes by share allocation (`rshare`).

u (Usage) Sorts lnodes by relative decayed usage.

`-r` *rate* Sets display update interval to *rate* seconds. Default is 5 seconds.

`-s` *crit* Selects the criteria for lnode or user information that is to be displayed. The following options for *crit* are available:

a (All) Display all users or lnodes. This is the default.

g (Selected groups) Display only selected groups. Enter groups to be displayed on the command line after all options.

i (Selected IDs) Display only selected IDs. Enter the selected IDs on the command line after all options.

o (Only groups) Display only resource groups.

Column Descriptions

The following columns of data are found in one or more of the `shrview` displays. Where columns with the same name have different meanings dependent on the display type, separate descriptions are provided for each display type.

Column	Description
`Adj_a`	The amount of group adjustment introduced by the `mingshare` parameter. Values of 1.0 indicated no adjustment, and values greater than 1.0 indicate increased priority for group members.
`Adj_l`	The amount of adjustment introduced by the maxushare parameter to limit the effect of past usage. Adjustments greater than 1 increase priority, and adjustments less than 1 will decrease priority.
`Chld`	Number of descendant lnodes that are members of this group.
`Chrg%`	Percentage of total cumulative charges (for selected lnodes) attributed to this ID.

CPU%	Percentage of total CPU time (for selected lnodes) attributed to this ID during the sample period.
Eshr%	Percentage of machine resources to which this user is entitled (as determined by the allocation of shares to users and groups).
Flags	An octal representation of the lnode kl.l_flags field.
Muse	The sum of the memory in use by all process that map to this lnode, expressed as clicks.
Name	Lnode user name, account, or group, as defined in the user database, with indentation to indicate hierarchy.
New%	(ADJGROUPS display only) The relative priority as adjusted by only the mingshare parameter, or actual priority without adjustments from the maxushare parameter.
New%	(LIMSHARE display only) The relative priority as adjusted by only the maxushare parameter, or actual priority without adjustments from the mingshare parameter.
Nrun	Current value of lnode kl_nrun field. This value is an estimate of the maximum amount of the rshare value (machine shares) that can be used by all runnable processes under the lnode.
Pri%	Actual priority for this ID. The priority value is relative to other selected lnodes and represents the percentage of machine resources that would be used by this ID, if all selected IDs were compute bound.
Proj%	Projected priority for this ID without the effect of the LIMSHARE or ADJGROUPS algorithms. The priority value is relative to other selected lnodes and represents the percentage of machine resources that would be used by this ID if all selected IDs were compute bound.
Rate	Current value of lnode kl_rate field. Rate is a decayed average of the number of runnable processes for this lnode.
Rate%	Percentage of total charges (for selected lnodes) ascribed to this ID during the sample period.
Ref	The number of processes mapped to this lnode, or, in the case of a group, the sum of all processes mapped to descendant lnodes.
Rshr%	Percentage of machine resources to which this user is entitled with the minimum value limited to the current value of the fair-share minimum parameter.
Svc	Service factor, an indicator of the balance between share allocation and past usage. The service factor is computed as Usg% / Shr%. A service factor greater than 1 indicates that this ID has recently received more than its allocated share and the fair-share priority will be low. A service factor less than 1 indicates that recent usage has been lower than allocated share and priority should be high.
Usage	The value of the lnode kl_usage field (represented as 1000s).
Usg%	Percentage of total decayed usage (for selected lnodes) accumulated by this ID.

Field Descriptions

The following are descriptions of fields in the statistics and nice displays.

Field	Description
`Active IDs`	The number of active IDs in the system and the maximum number of IDs
`Active groups`	The number of active groups in the system
`Usage`	The high water mark for usage values and the limit (`MAXUSAGE`)
`Share_pri`	The high water mark for `p_sharepri` values and the limit (`MAXUPRI`)
`Charge`	The percentage of total cumulative charges that are attributed to each of the cost factors
`Costs`	Current cost setting for each cost factor
`Counts`	Count of charges made for each of the cost factors
`N`	Process nice value
`Nice`	Plus or minus offset from normal nice (20)
`NiceDecays`	Priority decay rate and half-life in seconds for each nice value
`NiceRates`	A constant for each nice value at which the process rate (`kl_rate`) is incremented for each runnable process
`NiceTicks`	Rate at which a tick of CPU usage is charged for each nice value

NOTES

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
system, secadm	In a privileged administrator shell environment, allowed to write shell-redirected output to any file.
sysadm	Shell-redirected output is subject to security label restrictions.

If the PRIV_SU configuration option is enabled, the super user can write shell-redirected output to any file.

The shrview utility replaces the functionality of the shrates, shrinfo(1), shrstats(1), and shrusage(1) display commands, which are not available in the UNICOS 9.0 release.

EXAMPLES

The following example shows the use of shrview with the ADJGROUP display (-da option):

```
% shrview -da

  SHRVIEW  Type:adjgroup  Select:only groups  Sort_opt:id

     Name       Rate%   CPU%  Rshr%   Nrun  Rate  Proj%    Adj_a   New%   Pri%
------------  -----  -----  -----  -----  ----  -----  -------  -----  -----
 CCN            0.00   0.00  26.67   0.00  0.00  20.92     1.00  16.63  15.67
  SysAdm        0.00   0.00  17.78   0.00  0.00  13.82     1.00  10.99  10.35
  Syssup        0.00   0.00   8.89   0.00  0.00   7.10     1.00   5.64   5.32
 Mktg           0.31   0.23  13.33   0.00  0.00  56.65     1.00  45.05   9.99
  Country       0.09   0.00   4.44   0.00  0.00  53.91     1.00  42.87   6.66
  Intl          0.22   0.23   4.44   0.00  0.00   0.83     1.00   0.66   1.66
  TechOps       0.00   0.00   4.44   0.00  0.00   1.91     1.00   1.52   1.66
 SoftDev       97.62  99.77  60.00  65.76 21.48  22.42     1.00  38.31  74.34
  Userint      12.37  21.14  10.00  14.00 12.31   3.73     1.00   2.97   9.05
  Users         3.29   7.95  10.00  11.76  1.94  11.30     1.00  22.09  18.66
   Netdev       1.59   3.86   2.00   2.00  1.00   0.00     1.00   0.00   0.75
  Xydev        81.96  70.68  40.00  40.00  7.22   7.39     1.00  13.25  46.63
```

FILES

/usr/include/sys/share.h	Definition of shconsts structure (see share(5))
/usr/include/sys/lnode.h	Definition of lnode structure (see lnode(5))

SEE ALSO

`lnode`(5), `share`(5) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

`shradmin`(8), `shrmon`(8), `shrtree`(8) in the *UNICOS Administrator Commands Reference Manual*, Cray Research publication SR–2022

UNICOS Resource Administration, Cray Research publication SG–2302

NAME

`sim` – Invokes an interactive Cray simulator

SYNOPSIS

`sim` [`-d`] [`-i` *dfile*] [`-m` *n*] [`-t`] [`-u`] [`-v`] [*command* [*args*]]

IMPLEMENTATION

Cray PVP systems

DESCRIPTION

The `sim` utility invokes an interactive Cray simulator. It can simulate user programs for the Cray PVP systems. The simulator builds an argument vector when the program is loaded.

The `sim` utility accepts the following options:

`-d` Starts the simulator with display of tracing off.

`-i` *dfile* Reads simulator directives from *dfile*.

`-m` *n* Sets the debug message level to *n*.

`-t` Turns on instruction timing.

`-u` Turns on vector and functional use display. Timing and trace display must also be on.

`-v` Turns off virtual memory. Usually, the simulator keeps the entire absolute file in memory but uses virtual memory pages for any BSS space and any space that the simulated program requests. This option tells the simulator to keep the entire simulated image in memory.

command Names an absolute binary file created by the loader.

args Specifies arguments for the simulated program.

Commands

Commands to `sim` are single characters followed by other optional information. Commands fall into several categories:

- Execution control (`x22`)
- Display control (`.a=100`)
- Setting memory and registers (`s0=123`)
- File manipulation (`f=file`)

The following is a summary of the available commands.

`^C` <`CONTROL-c`> interrupts simulation and prompts for more commands.

`^D` <`CONTROL-d`> causes the last command to be repeated. This is especially useful for stepping through a program several lines at a time.

`![`*command*`]` Executes shell commands. If no argument is specified, the simulator starts a shell, and suspends until the shell reads an end-of-file. If *command* is supplied, the simulator spawns a shell to execute the command.

`#[`*n*`[=`*c*`]]` Controls window displays. This command is usable only if you are using a workstation that is running the X Window System environment. Up to 10 display windows are available. The parameter *n* is a single-decimal digit indicating the window number. The parameter *c* is a single character that is the display name to use (see the following display controls). If # is entered alone, all current windows are listed. If no display indicator is specified, the window is terminated. The simulator updates all display windows whenever it displays a prompt and every 2 seconds when running.

`*[`*comment*`]` Allows commenting. This command is especially useful when you are using an alternative command file.

`?` Displays available commands and their syntaxes.

`+[`*n*`]` Scrolls the last display forward *n* octal words. The default is the display size.

`-[`*n*`]` Scrolls the last display backward *n* octal words. The default is the display size.

`.[`*a*`[=[`*add*`][`*type*`][,`*size*`][`*format*`][/`*label*`]]]`

Controls displays. There are 26 displays available. They are named `a` through `z` and can be set up individually. The displays are saved with checkpoints for convenience. A period (`.`) specified alone causes the current display setup to be printed. A period followed by a single letter (*.a*) causes that display to be printed. The `.p` display is special cased to always follow the current program address.

The *add* variable can be a global symbol, an octal address, or an A or S register indicator (for example, `!a0` uses the contents of register `A0`).

The *type* variable can be one of the following:

- `b` B register
- `c` Common memory
- `S` Shared registers
- `t` T register
- `x` Exchange package
- `v` Vector register (for example, `120v` is register `V1`, element `020`)

The *size* variable is the octal number of words to print for the display.

The *format* variable is the desired display format. The format can be one of the following:

- `b` Bit format
- `B` Byte format
- `d` Decimal format
- `f` Floating-point format
- `h` Hexadecimal format
- `i` Instruction format

p Parcel format
s Short integer format
w Word format

The optional *label* parameter allows the display to be labeled. This labeling has no effect other than to be displayed when listing displays.

/*symbol*[+*offset*][=*value*]
Displays the location of *symbol* (plus an optional *offset*) or optionally changes the contents of that location. This value may be a string of octal digits optionally followed by a parcel indicator (a, b, c, d). It may also be a quoted string (for example, 'ABC') that is entered right-adjusted, zero-filled.

<*file* Opens *file* and reads directives from it. Continues reading from stdin when an end-of-file is reached. The directive file may also specify another directive file, but there is no return to the original directive file.

>[>][:]*file* Opens trace file. This command creates the named file and begins duplicating all output onto this file. If the second > symbol is specified, all output will be appended if the file already exists. If the colon (:) is specified, instruction traces will only be written to the trace file and not to the terminal.

add[*p*]=*val* Stores a value *val* in memory address *add*. This value may be a string of octal digits optionally followed by a parcel indicator (a, b, c, d). It may also be a quoted string (for example, 'ABC') that is entered right-adjusted, zero-filled. A parcel indicator *p* may also be appended to the address to show that only one parcel is to be changed.

a*n*[=*val*] Sets or displays A register *n*. If *val* is specified, it is an octal value.

br*n*=*val* Sets B register *n*. This value may be a string of octal digits optionally followed by a parcel indicator (a, b, c, d). It may also be a quoted string (for example, 'ABC') that is entered right-adjusted, zero-filled.

b[*n*[=*padd*[/*label*][(*cond*)][;*cmds*]]]
Controls breakpoints. If b is entered alone, all current breakpoints are listed. If no parcel address is specified, the breakpoint is cleared. The parameter *n* is a single decimal digit.

The *padd* variable may be an octal parcel address (for example, "52b"), a global symbol name, or an A or S register indicator (for example, "!a0").

The *cond* variable is a condition expression. Conditional breakpoint expressions are of the form (*operand operator operand*). *operand* may be a common memory location (for example, (100) or (symbol)), an A or S register (for example, a4 or s7), or a constant. Memory addresses and constants are assumed to be octal. *operator* may be either: =, !=, <, <=, >, or >=.

The *cmds* variable is an optional string of commands, separated by semicolons, to be executed when the breakpoint is reached. These may include setting other breakpoints (this is where using a register as a breakpoint address comes in handy). This provides for a maximum of 10 active breakpoints; breakpoints remain active until cleared (breakpoint must be cleared before it is reused). The optional *label* parameter allows the breakpoint to be labeled. This labeling has no effect other than to be displayed when listing breakpoints and when the breakpoint is hit.

`c=`*file* Creates a checkpoint of the current simulator state in *file*. The file is overwritten if it already exists. All breakpoints and display setups are recorded in the checkpoint file.

`d{+,-,r}` Controls display of instruction tracing. The `+` turns on all tracing, `-` turns off all tracing, and `r` turns on tracing of only return jumps.

`f=`*file* [*args*] Loads the absolute binary *file*. This file is the output from the loader. *args* are the arguments for the simulated program.

`h` Displays a history, which is also known as a traceback.

`i<`[*file*] Redirects the simulated program's input from a file other than the terminal. If *file* is not specified, the input is read from the terminal.

`i{b,i[=0],j,o,p,t,v}`
Prints information about simulation. The following options are recognized:

- `b` Instruction buffer information (timing must be on).
- `i` Number of times each instruction was executed; the optional `=0` allows you to zero these instruction counts.
- `j` Information about conditional jump execution.
- `o` Information about various types of operations.
- `p` Virtual memory page information.
- `t` Simulator time used.
- `v` Vector memory stride statistics.

`m=`*n* Sets the debug message level to *n*. The higher the message level, the more debug messages printed.

`o>`[`>`[*file*]] Sends the standard output of the simulated program to another file. If the second `>` is specified, the data is appended. If *file* is not specified, the output is directed back to the terminal.

`p=`*padd* Sets the P register to a parcel address. *padd* may be a global symbol or an octal parcel address.

`pn=`*n* Changes the currently active processor number to processor *n*. The simulator provides the capability to simulate multitasking programs. The `_tfork`(2) system call activates another simulated processor and makes a copy of the current register and local memory contents. The simulator automatically switches between processors whenever a semaphore is cleared or when a semaphore was tested but already set. This command allows switching on demand. The register and local memory displays always display the contents of the currently active processor.

`q` Terminates the simulator (quit).

`r=`*file* Restarts the simulation from a checkpoint in *file*.

`s`*n*`=`[*val*] Sets or displays S register *n*. This value may be a string of octal digits optionally followed by a parcel indicator (`a`, `b`, `c`, `d`). It may also be a quoted string (for example, `'ABC'`) that is entered right-adjusted, zero-filled.

`sb`*n*`=`*val* Sets shared B register *n*. This value may be a string of octal digits optionally followed by a parcel indicator (`a`, `b`, `c`, `d`). It may also be a quoted string (for example, `'ABC'`) that is entered right-adjusted, zero-filled.

`st`*n*`=`*val* Sets shared T register *n*. This value may be a string of octal digits optionally followed by a parcel indicator (`a`, `b`, `c`, `d`). It may also be a quoted string (for example, `'ABC'`) that is entered right-adjusted, zero-filled.

`T`[*n*[`=`*func*]] Specifies a timing range. A timing range is an address range within which the simulator keeps track of the total time spent executing the simulated program. Instruction timing must be on for range timing to be in effect. If `T` is entered alone, all current timing ranges are listed with their current totals. If no function name is specified, the timing range is cleared. The parameter *n* is a single decimal digit.

The parameter *func* is the name of a function in the simulated program. It is used as the start address of the timing range. The simulator sets the end address of the range to the address of the next closest function. This mechanism provides for a maximum of 10 active timing ranges. Timing ranges remain active until cleared. A timing range must be cleared before it is reused.

`t{+,-}` Controls instruction timings. The `+` turns on instruction timings. It also zeroes the timer if timing was already on. The `-` turns off instruction timings.

`tr`*n*`=`*val* Sets T register *n*. This value may be a string of octal digits optionally followed by a parcel indicator (`a`, `b`, `c`, `d`). It may also be a quoted string (for example, `'ABC'`) that is entered right-adjusted, zero-filled.

`u{+,-}` Controls vector and functional unit use display. The `+` turns on the use display. The `-` turns off the use display. Instruction trace and timing must also be on. The simulator lists the vector registers and functional units that are busy along with the instruction trace. The following abbreviations are used:

0–7 = Vector registers in use

M = Floating-multiply unit in use
A = Floating-add unit in use
R = Floating-reciprocal unit in use
S = Vector-shift unit in use
I = Vector-integer unit in use
C = Common memory in use

Vn[*/format*] Displays contents of vector register *n* in format *format*.

vn.elem=val Sets V register *n*, element *elem* to a value *val*. This value may be a string of octal digits optionally followed by a parcel indicator (a, b, c, d). It may also be a quoted string (for example, 'ABC') that is entered right-adjusted, zero-filled.

vl=*val* Sets the vector length register to a decimal value *val*.

vm=*val* Sets the vector mask register to an octal value *val*.

w[*n*[=[*add1*][,*add2*][{r,w}][/*label*][(*cond*)]]]

Controls watchpoints. Watchpoints are memory addresses that are watched for a reference. If the specified address or range of addresses is referenced, the simulator stops and prints a message. If w is entered alone, all current watchpoints are listed. If no address is specified, the watchpoint is cleared. The parameter *n* is a single decimal digit.

add1 is the start address of the area to be watched. *add2* is the end address of the area to be watched. If omitted, it is the same as the start address. The optional trailing letter gives the ability to watch for only reads or only writes.

If r is specified, only memory reads are watched. If w is specified, only memory writes are watched. The default is to watch both reads and writes.

The optional *label* parameter allows the watchpoint to be labeled. This has no effect other than to be displayed when listing watchpoints and when the watchpoint is hit.

cond is a condition expression. Condition expressions are the same as for breakpoints. This mechanism provides for a maximum of 10 active watchpoints.

Watchpoints remain active until cleared. A watchpoint must be cleared before it is reused.

x[{*n*,*,e,j,R,r,s}][{+,–}]

Executes instructions. A count *n* may be specified. The default is to execute one instruction. The x command may be followed by an optional letter, which indicates a condition for stopping execution. The following options are recognized:

* Indicates infinity
e Executes to the next EXIT instruction
j Executes to the next jump instruction (this includes return jumps and EXITs)
R Executes to the return to the caller of the current routine
r Executes to the next return jump instruction
s Executes to the next process switch

The optional trailing + or - is available to override the current instruction trace status. If the - is used when instruction tracing is currently on, the simulator executes silently to the next stopping point, but tracing still remains on by default.

FILES

`/usr/bin/sim`	Cray simulator
`.simrc`	Simulator configuration file

SEE ALSO

`_tfork`(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

NAME

`size` – Prints section sizes of executable files

SYNOPSIS

`size` [`-o`] [`-x`] *file* ...

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `size` utility produces size information in 64-bit words for each section in an absolute binary executable file. The size of the text, data, and BSS (uninitialized data) sections are printed along with their sum.

Numbers are printed in decimal, unless either the `-o` or `-x` option is used, in which case numbers are printed in octal or in hexadecimal, respectively.

The `size` utility accepts the following options and operand:

`-o` Prints numbers in octal. By default, numbers are printed in decimal.

`-x` Prints numbers in hexadecimal. By default, numbers are printed in decimal.

file ... Specifies one or more absolute binary executable files.

NOTES

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
`system`, `secadm`	Allowed to print size information for any executable file. In a privileged administrator shell environment, shell-redirected I/O is not subject to file protections.
`sysadm`	Allowed to print size information for any executable file subject to security label restrictions. Shell-redirected I/O is subject to security label restrictions.

If the `PRIV_SU` configuration option is enabled, the super user is allowed to print size information for any executable file. Shell-redirected I/O on behalf of the super user is not subject to file protections.

EXAMPLES

This example lists the text, data, and BSS space of the `a.out` file:

```
$ size a.out

a.out:  63070 + 19023 + 15240 = 97333
```

SEE ALSO

`pascal`(1), `segldr`(1)

`cc`(1) in the *Cray Standard C Reference Manual*, Cray Research publication SR–2074
`f90`(1) in the *CF90 Commands and Directives Reference Manual*, Cray Research publication SR–3901

`a.out`(5) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

NAME

`sleep` – Suspends execution for a specified interval

SYNOPSIS

`sleep` *time*

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The `sleep` utility suspends execution of a process until the number of real-time seconds specified by the *time* operand have elapsed.

The `sleep` utility supports the following operand:

time A non-negative decimal integer specifying the number of seconds for which to suspend execution.

NOTES

This utility can produce error output.

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
`system`, `secadm`	In a privileged administrator shell environment, allowed to write shell-redirected output to any file.
`sysadm`	Shell-redirected output is subject to security label restrictions.

If the `PRIV_SU` configuration option is enabled, the super user can write shell-redirected output to any file.

EXIT STATUS

The `sleep` utility exits with one of the following values:

0 The execution was successfully suspended for at least *time* seconds, or a `SIGALRM` signal was received.

>0 An error occurred.

EXAMPLES

Example 1: The following example uses `sleep` to execute a command after a certain amount of time:

```
(sleep 105; command)&
```

Example 2: The following shell script fragment tests for user `joe` every 10 seconds. When `joe` logs in, the script breaks out of its loop.

```
while true
do
        echo "entering loop"
        if who | grep joe > /dev/null
        then
                break
        else
                sleep 10
        fi
done
```

SEE ALSO

`alarm`(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

`sleep`(3C) in the *UNICOS System Libraries Reference Manual*, Cray Research publication SR–2080

NAME

sm – Invokes the UNICOS source manager (USM)

SYNOPSIS

sm [-V] [*USM_subcommand*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The UNICOS source manager (USM) is a source control system which runs under the UNICOS operating system. USM stores text and control information in USM structures called *program libraries* (PLs). When USM subcommands are run, USM uses the PLs in the current directory, unless an absolute path name is specified. For convenience, use the environment variable SMPREFIX to define the location of the PLs.

USM subcommands can be entered on the command line (*batch mode*), or USM can be invoked interactively (*interactive mode*).

In interactive mode, sm prompts for the SMPREFIX value, if undefined, and then prompts for the name of the PL on which you want to work. Thereafter, every subcommand entered will refer to this PL in the location specified by SMPREFIX. You can use the set subcommand to change either of these two values.

-V Displays the version of the sm command being used. Following is an example of the format:

```
Unicos Source Manager, Release 8.1
Change Date = 12/14/93 11:43:35
```

USM Subcommands

This subsection describes each USM subcommand and its options. The batch synopsis is given; the interactive synopsis is identical except that neither the initial sm command nor the PL is specified. For more information on USM use, see the *UNICOS Source Manager (USM) User's Guide*, Cray Research publication SG–2097.

The USM subcommands are listed alphabetically by subcommand name. Subcommand options are listed alphabetically except when they must be supplied; then the mandatory options are listed first.

sm add [-A *attribute*] [-b] [-B] [-n] [-N *named_branch*] [-o] [-r *VID*] [-z] *pl mods*
sm add -i *list* [-A *attribute*] [-b] [-B] [-n] [-N *named_branch*] [-o] [-r *VID*] [-z] *pl*

Adds a mod to the named USM PL. In interactive mode, *pl* is not specified. The add subcommand verifies that none of the files changed by the mod being applied are locked by another user through the -e option on the get subcommand. If locks are enabled, add also verifies that each file changed by the mod is locked by the current user.

If more than one mod is added with one command, the mods are treated as a single entity. Each mod must work; if not, all mods are removed.

`-i` *list* Specifies a file, *list*, that contains a list of mod names to be added to the PL. The file *list* can have a maximum of 500 entries.

`-A` *attribute* Applies the mod to the version of the file with the specified attribute.

`-b` Disables branching; specifies that a mod does not create a sideline branch.

`-B` Applies the mod to the most recent change on the sideline branch of the version specified.

`-n` Specifies retention of the edited file. (This file is usually removed when the lock is removed from the file after the mod is applied.)

`-N` *named_branch*
Applies the mod to the specified sideline branch.

`-o` Overrides lock restrictions. When locks are maintained on the PL, overrides the necessity to check out a file with intent to change before a mod can be added, or overrides a lock placed on the file by another user with the `get -e` subcommand. Only the owner of the PL can use the `-o` option.

`-r` *VID* Specifies the version ID (VID) to which the mod should be applied. This option is valid only when multiple versions of a PL are checked out.

`-z` Specifies the path name of the directory that contains the locked files changed by the added mods. If the `-z` option is not used, these files must be in the current directory.

pl (Batch mode only) The PL to which the command applies.

mods The mods to be applied to the PL. This argument is specified only when the `-i` option is not used.

`sm admin` [`-a` *login*] [`-A` *attribute*] [`-B`] [`-d` *flag* [*value*]] [`-e` *login*] [`-f` *flag* [*value*]] [`-l`] [`-L`] [`-N` *named_branch*] [`-o` *type*] [`-P`] [`-r` *VID*] [`-s`] [`-S` *VID*] [`-t`] [`-u`] [`-U`] [`-v` *VID*] [`-V`] [`-z`] *pl* [*files*]

Administers PLs. In interactive mode, *pl* is not specified. Only the owner of the PL (also referred to as the *administrator* of the PL) may use this subcommand.

`-a` *login* Adds a login name or group ID to the list of users who may make deltas (changes) to the PL. The argument *login* may be a comma-separated list of login names or group IDs. If *login* is preceded by a ! character, the specified user or group is denied permission to make changes to the PL. If the list is empty, a user with the correct access permissions to the PL is able to perform any action on the PL. In the case of ambiguities (for example, if the system has both a user ID and a group ID with the same name), USM's behavior is undefined.

`-A` *attribute* Performs the operation on the version of the module that has the specified attribute. Specifying `admin -A` *attribute* `-dA` is the the same as specifying `admin -dA`*attribute*.

`-B` Specifies that the command is to be applied to the most recent change on the sideline branch of the version specified.

`-d` *flag* [*value*]]
Disables *flag* for the PL. The following flags are available:

`A`[*attribute*] Removes the definition of the specified attribute for the modules listed. If no attribute is specified, removes all attributes from the specified version. Specifying `admin -A` *attribute* `-dA` is the the same as specifying `admin -dA`*attribute*.

`b` Allows only the PL owner (administrator) to create sideline branches.

`d` Disables the default sideline branch.

`h`[*name*] If *name* is given, removes that category from the mod header file currently in force for the PL. If *name* is omitted, disables the mod header file in force for the PL.

`i` Does not check for any string in modules in the PL.

`L` Disables hard module locks.

`m` Disables the automatic mod name generator.

`N`[*named_branch*]
Removes the definition of the specified sideline branch for the modules listed. If no sideline branch is specified, removes all sideline branches from the specified version.

`p` Disables the special substitution of the M keyword.

`s` Reverts to default mode (00444) for the source tree.

`S` Disables advisory locks.

`-e` *login* Erases a login name or group ID from the list of users who may make deltas (changes) to the PL. The argument *login* may be a comma-separated list of login names or group IDs.

`-f` *flag*[*value*]
Enables *flag* for the PL. The following flags are available:

`A`*attribute* Sets the specified attribute for the specified VID.

`b` Allows all valid PL users to create sideline branches.

`d`[*named_branch*]
Disables the sideline branch, reverting to the mainline as the default.

h[*filename*] Enables the specified mod header file; *filename* contains a description of the categories that are to go in the delta summary for the mods. If *filename* is not supplied, the `admin` command prompts interactively for the categories.

i*string* Enforces checking for *string* in each module in the PL. For example, if `-fi"%Z% %U%"` is specified, each module must contain the USM ID keywords `%Z%` (the *what line*) and `%U%` (the creation time). For more information on USM ID keywords, see the *UNICOS Source Manager (USM) User's Guide*, Cray Research publication SG–2097.

L Specifies hard module locks.

m[*no*] Sets the automatic mod name generator; *no* is optional and specifies a starting number.

N*branch* Sets the specified sideline branch for the specified VID.

p*string* Uses *string* instead of the PL name when substituting the M keyword.

s*mode* Sets the mode to be used in the source tree.

S Specifies advisory locks.

`-l` Locks the PL. This is useful when performing administrative tasks, to ensure that no user can access the PL.

`-L` Unlocks the PL, if it was locked with the `admin -l` subcommand.

`-N` *named_branch*
Performs the operation on the current version (specified by the `-A`, `-N`, or `-r` options) of the specified sideline branch.

`-o` *type* Specifies the type for updating a PL format.

`-P` Generates the specified files in the parallel source tree.

`-r` *VID* Specifies the VID of the file to which this subcommand applies.

`-s` Adds files (specified by the *files* argument) to the parallel source tree. This option is meaningful only if the PL does not maintain a full parallel source tree.

`-S` *VID* Creates a stub version for the named modules.

`-t` When regenerating the source tree, sets the time stamp for each updated file to the initial creation time of the VID. By default, the time stamp is set to the current time.

`-u` Removes files (specified by the *files* argument) from the parallel source tree. This option is meaningful only if the PL does not maintain a full parallel source tree.

`-U` Converts a PL from an old USM format to the next one. If the current PL type is "`d`", it will be updated to "`e`". If the current PL type is "`e`", it will be updated to "`f`".

`-v` *VID* Specifies the next VID for all modules to the PL. The next time a module is modified on the mainline, it will be set to this VID.

`-V` Changes all named files to the default VID. This forms a new major version; mods added before this new major version cannot be deleted. A file or a directory name must be specified with the `-V` option. If a directory name is specified, all files in that directory are included in the new major version. If a `.` symbol is specified, all files in the PL are included.

`-z` Recalculates the checksum and VIDs for the specified files.

pl (Batch mode only) The PL to which the `admin` subcommand applies.

files The individual files to which the `admin` subcommand applies. If *files* is a directory in the PL, each file in that directory is processed. If *files* is a `.` symbol, each file in the PL is processed.

`sm create` [`-l`] [`-O` *type*] [`-s`] [`-S`] [`-t`] [`-v` *VID*] *fromdir todir*

Creates a PL from *fromdir*. In interactive mode, you are prompted for the name of the PL. If *fromdir* is a PL that has UPDATE directives embedded in the code, or if it is a directory of UPDATE source files, the deck names are taken from the PL. This puts some limitations on the use of USM by users and the `-s` option of `create` must be used.

If *fromdir* is not an `UPDATE` PL, it must be a directory name. Each file in the named directory is read as input. Existing `.USM` and `.SCM` directories are ignored, which allows creation from existing USM PLs.

For a group PL, all users must have read access to the PL. This is often accomplished by creating a group for the users, then executing the `chgrp`(1) command on the PL and its contents to grant read access to the group. Another option is to create an interface, or *setuid*, program to allow access to the PL without requiring too-liberal permissions. For more information on creating an interface program, see the *UNICOS Source Manager (USM) User's Guide*, Cray Research publication SG–2097.

`-l` Creates a `locks` file. This enables the PL to lock files that are checked out with intent to change.

`-O` *type* Creates a specific format PL type. The *type* values must be "`d`", "`e`", or "`f`".

`-s` Indicates that the *fromdir* directory contains source files that have UPDATE directives already in them, and that the deck names are to be taken from the source files. If the `-s` option is used, no parallel source tree is maintained.

`-S` Enables advisory locks.

`-t` Does not create a parallel source tree.

`-v` *VID* Specifies the VID to be assigned to all files in the PL. This VID becomes the default version ID for all subsequent files added to the system.

fromdir The directory that is an `UPDATE` PL or that contains source files to be read by `create`. If *fromdir* is a `.` symbol, each file in the directory is processed.

todir The name of the new PL to be created.

`sm delete` [`-l`] [`-o`] *pl mod*

Deletes the specified mod from a PL. In interactive mode, *pl* is not specified. The `delete` subcommand is restricted to the owner of the PL.

The `delete` subcommand evaluates for dependencies all mods that are applied after the specified mod is applied. If dependencies are found, the specified mod cannot be removed. If the file is currently locked by a user, `delete` fails.

When deleting a mod on a sideline branch, the mod must be a terminal node; that is, it must be the last mod applied to that branch. A mainline mod that is the start of a sideline branch has that branch as a dependency; that mod cannot be deleted.

If the PL has been updated to a new major version number (that is, if the `admin -V` subcommand has been run on the PL) since the specified mod was added, the mod cannot be deleted without specifying the `-o` option.

`-l` Specifies that the operation to delete a mod can succeed only if the specified mod is the last mod applied to all affected modules.

`-o` Overrides lock restrictions. If locks are maintained on the PL and the mod being removed changes a file currently locked by another user, `-o` removes the lock on the file and deletes the mod anyway. Only the owner of the PL can use the `-o` option.

pl (Batch mode only) The PL to which the command applies.

mod The mod to be deleted from the PL.

`sm delta -m` *modid* [`-a`] [`-A` *attribute*] [`-b`] [`-B`] [`-e` *string*] [`-f`] [`-F`] [`-k` *string*] [`-K` *file*] [`-n`] [`-N` *named_branch*] [`-r` *VID*] [`-t` *name*] [`-u`] [`-v`] [`-y` *comment*] [`-z`] *pl files*

Makes and applies a delta (mod) to a PL. In interactive mode, *pl* is not specified.

The `delta` subcommand creates a mod to each named file, applies it to the specified PL, and returns the resulting mod to the user, in a file named *modid*. The `delta` subcommand verifies that the user who is attempting to create the delta is the same user who has the file reserved (through the `-e` option to the `get` subcommand). If the specified file does not exist, but file *file*`.u` does exist, `delta` adds the file to the PL (that is, it submits a mod to add the file to the PL).

`-m` *modid* Specifies the name of the mod to be produced. (This option must be specified unless the automatic mod name generation flag is set for the PL by using the `-fm` option of `admin`.) When `delta` completes, the mod is in a file named *modid*.

`-a` Specifies that the alternate table entries should be used when generating the mod. The table entries determine the characteristics of the mod. The alternate tables perform optimization based on different criteria from the default tables; using this option might produce a mod that is more optimal.

`-A` *attribute* Applies the mod to the version of the file that has the specified attribute.

`-b` Disables branching; specifies that the mod does not create a sideline branch.

`-B` Applies the mod to the most recent change on the sideline branch of the version specified.

`-e` *string* Specifies a string to use as the pattern when unexpanding keywords in PLs that do not support enforced keywords.

`-f` Forces the creation of a mod for all modules, whether or not they have been modified.

`-F` Allows a flexible mod header. By default, when processing information supplied with the `-k` and `-K` options, all header fields must be defined in the `USMHEADER` file. Using the `-F` option overrides this restriction.

`-k` *string* Specifies a string containing the keywords for the mod header. If the `-k` option is not used, `delta` prompts for this information. The format of the mod header is specified for the PL by using the `-fh` option of the `admin` subcommand.

`-K` *file* Specifies the file that contains the keyword information for use in the mod header.

`-n` Specifies retention of the edited file (usually removed at completion of delta processing).

`-N` *named_branch*
Applies the mod to the specified sideline branch.

`-r` *VID* Specifies the version of the PL when more than one version is checked out.

`-t` *name* Specifies a file containing the comment text to be included in the mod. If neither the `-y` or the `-t` option is used, `delta` prompts for this comment text.

`-u` Unexpands the enforced string of keywords (set with the `admin` subcommand) before making the mod.

`-v` Verbose mode. Issues messages about unchanged modules.

`-y` *comment* Specifies the comment text to include in the mod. If the `-y` option is not used, `delta` prompts for this comment text.

`-z` Uses full path names when searching for the named files. Usually, `delta` searches for the named files in the current directory.

pl (Batch mode only) The PL for which the delta is to be created.

files The files for which the mod is to be made. If *files* is a directory in the PL, each file in that directory is taken as an input file. If *files* is a `.` symbol, each file in the PL is processed. If *files* is a `-`, the list of files to process is read from standard input.

`exit`

(Interactive mode only) Exits from interactive mode.

`sm get` [`-a` *attribute*] [`-A` *attribute*] [`-b`] [`-B`] [`-c`] [`-d` *option*] [`-D`] [`-e`] [`-E`] [`-g`] [`-h`] [`-i` *VIDs*] [`-I`] [`-k`] [`-l`] [`-m`] [`-M` *mod*] [`-n`] [`-N` *named_branch*] [`-p`] [`-q`] [`-r` *VID*] [`-R` *VID*] [`-s`] [`-t`] [`-T` *date*] [`-U`] [`-v`] [`-w`] [`-x` *VIDs*] [`-z`] *pl files*

Retrieves a version of the specified files from the PL. In interactive mode, *pl* is not specified. For most options, this subcommand generates an ASCII text file from each specified file in the specified PL, according to the arguments given.

The most common option is `-e`; it returns a file that can be changed and then added to the PL as a mod with the `delta` subcommand or the `add` and `mod` subcommands. If the `-e` option is used, `get` returns two files. If the PL is set up to be pure text, *file* and *file*`.u` are returned; *file* does not need any further processing to be compiled or edited. If the PL is set up to contain `UPDATE` directives, *file*`.e` and *file*`.u` are returned; *file*`.e` must be further processed by `nupdate`(1) before it can be compiled or edited. The *file*`.u` file has a mode of 440; this is a binary control file that must not be changed or removed.

The `-n`, `-m`, `-I`, `-h`, and `-p` options provide control information in a copy of the file. For a file that supports `UPDATE` directives, there are many restrictions on these options, see the *UNICOS Source Manager (USM) User's Guide*, Cray Research publication SG–2097.

`-a` *attribute* Specifies the attribute of the version on which the delta report is based.

`-A` *attribute* Returns the version that has the specified attribute defined.

`-b` Specifies an operation on a sideline branch, if one does not yet exist. (If a sideline branch already exists, use `-r`*VID*.) This option increments the branch number of the VID.

`-B` Returns (in *file*`.d`) the most recent change on the sideline branch of the specified version.

`-c` Produces a complete report on the module when processing the `-l` and `-h` options. By default, only lines that were part of the specified VID are included in the report.

`-d` *option* Defines an option to pass on to `nupdate`(1). Use this option to define identifiers that extract a particular configuration.

`-D` Returns a delta report (in *file*`.d`) that indicates what changed after the specified version. The report indicates which lines are new and which have been deleted.

`-e` Indicates that the `get` subcommand is used to edit or make a change (mod) to the PL, followed by use of the `delta` subcommand or the `add` and `mod` subcommands. If locks are enabled, the `-e` option places a lock on the PL to prevent other `get` operations from editing the same VID. The lock remains until changes are checked in (with the `delta` or `add` subcommand), or until the the `unget` subcommand is used to return the file unchanged.

`-E` Retains the effective user ID when returning files. The default is to return files owned by the real user. (This option is useful only for effective user ID interface programs.)

`-g` Suppresses the actual retrieval of text from the USM file. It is used with the `-l` option or to verify the existence of a particular VID.

`-h` Shows each line that exists in the named files. Places the characters `<i><TAB>` at the start of any line that has been deleted from the current version of the file; each line in the retrieved file (including current and deleted lines) then has the following format:

`<i><TAB>`*line*

This information appears after the information supplied by the `-n`, `-m`, and/or `-I` options, if used (`<TAB>` is a tab character).

`-i` *VIDs* Specifies a list of VIDs or modification IDs to include in the retrieved file. This option overrides any other VID specification.

`-I` Places the modification ID at the start of each line of the retrieved files, in the following format:

modid`<TAB>`*line*

The modification ID appears after the information supplied by the `-n` and/or `-m` options, if used, and before the information supplied by the `-h` option (`<TAB>` is a tab character).

`-k` Suppresses replacement of identification (ID) keywords and processing of `UPDATE` directives in the retrieved text by their value. The `-k` option cannot be used if the `-e` option is specified.

`-l` Causes a delta summary to be written into a file named *file*`.l`. The delta summary contains mod header information and user comments added when the mod was created.

`-m` Places the VID at the start of each line of the retrieved files in the following format:

VID`<TAB>`*line*

The VID appears after the information supplied by the `-n` option, if used, and before the information supplied by the `-I` and/or `-h` options. (`<TAB>` is a tab character.)

`-M` *mod* Specifies a list of mods applied to the USM file in the PL before the file is returned to the user. This is used to test either a mod received from the field or a mod written by hand.

`-n` Places the module name at the start of each line of the retrieved files, in the following format:

module_name<TAB>*line*

The module name appears before the information supplied by the `-m`, `-I`, and/or `-h` options, if used. (<TAB> is a tab character.)

`-N` *named_branch*
Specifies that the VID of the module to be retrieved is the current VID of the specified sideline branch.

`-p` Writes the retrieved files to standard output rather than to a file. Unless the `-s` option is also used, the messages produced by `get` are redirected to standard error.

`-q` Retrieves all files changed by the mods applied with the `-M` option. This option is extremely useful for testing a list of mods that affects a large number of files.

`-r` *VID* Specifies the VID of the mod that is to be retrieved.

`-R` *VID* Specifies the version on which the delta report is based.

`-s` Suppresses the redirection of messages produced by `get`. This option is valid only when the `-p` option is also used.

`-t` Sets the time stamp for the specified file to the time the delta was created. This option is useful for PLs manipulated with makefiles; only the changed files have new dates when accessed with `get -t`.

`-T` *date* Retrieves files as they existed on the specified date.

`-U` Retrieves a `.u` file.

`-v` Verbose mode. Continues processing all modules even when an error is encountered.

`-w` Returns files with write permission enabled, even if the `-e` and `-k` options are not specified.

`-x` *VIDs* Specifies a list of VIDs or modification IDs to exclude from the retrieved file. This option overrides any other VID specification.

`-z` Puts the retrieved file in the directory (specified by *files*) rather than in the current directory.

pl (Batch mode only) The PL to which the command applies.

files The files to which the command refers in the PL. If *files* is a directory, each file in the directory is retrieved. If *files* is a `.` symbol, all files maintained in the PL are retrieved. If *files* is a `-`, the list of files is read from standard input.

`sm help` [*subcommand*]

Supplies a helpful message about the specified USM subcommand. If no subcommand is specified, `help` supplies a general help message that includes a list and a brief description of all USM subcommands.

`sm history` [`-A` *attribute*] [`-B`] [`-c`] [`-N` *named_branch*] [`-r` *VID*] [`-T` *date*] *pl files*

Writes a history of the specified files to standard output. It lists the version number, the date each version was created, the mod that creates each version, and any attributes or sideline branches assigned. If *files* is a `.` symbol, each file in the PL is processed.

`-A` *attribute* Returns history for the VID with the specified attribute.

`-B` Returns information from the sideline branch that starts at the specified VID.

`-c` Specifies that the complete history should be returned. By default, only the history of the specified VID is returned.

`-N` *named_branch*
Returns history for the VID on the specified branch.

`-r` *VID* Specifies the VID.

`-T` *date* Returns the history of the modules since the specified date.

pl (Batch mode only) The PL to which the command applies.

files The individual files to which the `history` subcommand applies. If *files* is a directory in the PL, each file in that directory is processed. If *files* is a `.` symbol, each file in the PL is processed.

`sm list` [`-A` *attribute*] [`-b`] [`-B`] [`-N` *named_branch*] [`-r` *VID*] [`-T` *date*] *pl files*

Writes a list of files in the specified PL, with the current VID of each, to standard output. In interactive mode, *pl* is not specified. If *files* is a directory, a list of all files maintained in that directory is written. If *files* is a `.` symbol, each file in the PL is listed.

`-A` *attribute* Specifies the attribute of the VID to be displayed.

`-b` Display only the module names, not the VIDs.

`-B` Returns information from the sideline branch that starts at the specified VID.

`-N` *named_branch*
Specifies the sideline branch of the VID.

`-r` *VID* Specifies the VID to be displayed.

`-T` *date* List the modules as they existed on the specified date.

pl (Batch mode only) The PL to which the command applies.

files The individual files to which the list subcommand applies. If *files* is a directory in the PL, each file in that directory is processed. If *files* is a `.` symbol, each file in the PL is processed.

`sm merge -m` *modid* [`-A` *attribute*] [`-B`] [`-e` *string*] [`-f`] [`-F`] [`-h`] [`-i`] [`-k` *string*] [`-K` *file*] [`-l`] [`-N` *named_branch*] [`-q`] [`-r` *VID*] [`-t` *name*] [`-u`] [`-y` *comment*] [`-z`] *pl files*

Merges changes for out-of-date modules when advisory locks are used.

If `merge` finds your changes can be applied, it updates the lock to the current VID and returns the merged files with names *file* and *file*`.u`. The original files are returned with names *file*`,o` and *file*`,u`. A delta report is returned in *file*`.d`. You should carefully check the suggested merged file before checking it in by using the `delta` or `mod` subcommands.

`-m` *modid* Specifies the name of the mod to be produced. (This option must be specified unless the automatic mod name generation flag is set for the PL by using the `-fm` option of `admin`.) When `merge` completes, the mod is in a file named *modid*.

`-A` *attribute* Specifies that the version to lock is the one with the specified attribute.

`-B` Specifies that the version to lock is the most recent change on the sideline branch of the version specified.

`-e` *string* Specifies a string to use as the pattern when unexpanding keywords in PLs that do not support enforced keywords.

`-f` Forces the creation of a mod for all modules, whether or not they have been modified.

`-F` Allows a flexible mod header. By default, when processing information supplied with the `-k` and `-K` options, all header fields must be defined in the `USMHEADER` file. Using the `-F` option overrides this restriction.

`-h` Specifies that the mod to be created should not have a mod header.

`-i` Information mode. Displays the names of the out-of-date modules. No mod is generated when the `-i` option is used.

`-k` *string* Specifies a string containing the keywords for use in the mod header. If the `-k` option is not used, `merge` prompts for this information. The format of the mod header to be used is specified for the PL by using the `-fh` option of the `admin` subcommand.

`-K` *file* Specifies the file that contains the keyword information for use in the mod header.

`-l` Updates the source files only; the locks file (`.USM/locks`) is not updated.

`-N` *named_branch*
Specifies that the VID of the module to be locked is the current VID of the specified sideline branch.

`-q` Specifies quick mode. This option updates only the `locks` file; it does not produce a mod and does not merge files.

`-r` *VID* Specifies the VID created by this mod.

`-t` *name* Specifies a file containing the comment text to include in the mod. If neither the `-y` nor `-t` option is used, `merge` prompts for this comment text.

`-u` Unexpands the enforced string of keywords before making the mod.

`-y` *comment* Specifies the comment text to include in the mod. If neither the `-y` nor `-t` option is used, `merge` prompts for this comment text.

`-z` Specifies that the full path name is to be used in all files specified. If the `-z` option is not used, specified files must be in the current directory.

pl (Batch mode only) The PL to which the command applies.

files The individual files to which the merge subcommand applies. If *files* is a directory in the PL, each file in that directory is processed. If *files* is a `.` symbol, each file in the PL is processed.

`sm mod -m` *modid* [`-a`] [`-A` *attribute*] [`-b`] [`-B`] [`-e` *string*] [`-f`] [`-F`] [`-h`] [`-k` *string*] [`-K` *file*] [`-N` *named_branch*] [`-r` *VID*] [`-t` *name*] [`-u`] [`-v`] [`-y` *comment*] [`-z`] *pl files*

Creates a mod from two files: *file* and *file*`.u`. In interactive mode, the PL is not specified. (If the PL supports UPDATE directives, the two files are *file*`.e` and *file*`.u`.) This mod can be applied later to the PL with the `add` subcommand.

If the specified file does not exist in the current directory, but the *file*`.u` file does exist, `mod` creates a mod to purge the specified file from the PL. If the specified file exists, and the *file*`.u` file does not exist, `mod` creates a mod to add the specified file to the PL.

`-m` *modid* Specifies the name of the mod to be created. (This option must be specified unless the automatic mod name generation flag is set for the PL by using the `-fm` option of `admin`.) When `mod` completes, the mod is in a file named *modid*.

`-a` Specifies that the alternate table entries should be used when generating the mod. The table entries dictate the characteristics of the mod. The alternate tables perform optimization based on different criteria from the default tables; using this option might produce a mod that is more optimal.

`-A` *attribute* Ensures that the VID with the specified attribute is locked when checking locks.

`-b` Prohibits `mod` operation when the version locked is a sideline branch.

`-B` Ensures that the most recent sideline change of the file is locked when checking locks.

`-e` *string* Specifies a string to use as the pattern when unexpanding keywords in PLs that do not support enforced keywords.

`-f` Forces the creation of a mod for all modules, whether or not they have been modified.

`-F` Allows a flexible mod header. By default, when processing information supplied with the `-k` and `-K` options, all header fields must be defined in the `USMHEADER` file. Using the `-F` option overrides this restriction.

`-h` Specifies that the mod to be created should not have a mod header.

`-k` *string* Specifies a string containing the keywords for the mod header. If the `-k` option is not used, `mod` prompts for this information. The format of the mod header to be used is specified for the PL by using the `-fh` option of the `admin` subcommand.

`-K` *file* Specifies the file that contains the keyword information for use in the mod header.

`-N` *named_branch*
Ensures that the current VID on the named sideline branch is locked when checking locks.

`-r` *VID* Ensures that the specified VID is locked when checking locks.

`-t` *name* Specifies a file containing descriptive text to include as the initial comment in the mod. The `-t` and `-y` options are mutually exclusive.

`-u` Unexpands the enforced string of keywords before making the mod.

`-v` Verbose mode. Issues messages about unchanged modules.

`-y` *comment* Specifies the comment text to include in the mod header. The `-y` and `-t` options are mutually exclusive.

`-z` Specifies that the full path name is to be used in all files specified. If the `-z` option is not used, specified files must be in the current directory.

pl (Batch mode only) Specifies the PL to which the mod applies.

files Specifies the files for which the mod should be generated. If *files* is a directory in the PL, each file in that directory is processed. If *files* is a `.` symbol, each file in the PL is processed. If *files* is a `-` character, the list of files to process is read from standard input.

`params`

(Interactive mode only) Displays the value of `SMPREFIX` and the name of the PL that is being modified.

`sm query` [`-A` *attribute1* [,*attribute2*]] [`-b`] [`-B`] [`-d`] [`-f`] [`-i`] [`-I`] [`-l`] [`-m`] [`-M`] [`-N` *named_branch*] [`-r` *VID*] [`-s`] [`-T` *date*] [`-u` *user*] [`-v`] *pl files*

Returns information about the current state of the specified PL.

`-A` *attribute1* [,*attribute2*]
: Specifies to show information about the version with the specified attribute or attributes. If two attributes are specified, module history information is limited to the VIDs occurring between the two attributes.

`-b`
: Displays information in an abbreviated format.

`-B`
: Specifies to show information about the version with the most recent change on the sideline branch of the version specified.

`-d`
: Displays the named branch that is currently the main line.

`-f`
: Displays a list of the flags that are enabled in the PL.

`-i`
: Lists all modules that are inactive in the list.

`-I`
: Displays the table of all known identifiers in the PL.

`-l`
: Indicates the VIDs that are locked for each module.

`-m`
: Displays a list of the mod identifiers that make up the specified VID.

`-M`
: Displays only the modules if the `-I` option is used to display a table of known identifiers.

`-N` *named_branch*
: Specifies that the VID of the module to be listed is the current VID of the specified sideline branch.

`-r` *VID*
: Specifies the VID to which the command applies.

`-s`
: Excludes modification IDs on sideline branches when displaying the identifier list.

`-T` *date*
: Returns information about the VID that existed on the specified date.

`-u` *user*
: Limits the lock information displayed to the modules locked by the specified user.

`-v`
: Displays a list of the VIDs that make up the specified VID.

pl
: (Batch mode only) The PL to which the subcommand applies.

files
: The individual files to which the subcommand applies. If *files* is a directory in the PL, each file in that directory is processed. If *files* is a `.` symbol, each file in the PL is processed.

`quit`

(Interactive mode only) Exits from `sm`.

`sm recover` *pl*

(Batch mode only) Corrects problems caused by the failure of USM subcommands. The only subcommand that is not recoverable by `recover` is the `release` subcommand. The `recover` subcommand is restricted to the owner of the PL.

`sm release` [`-p`] [`-r` *VID*] *pl* [*files*]

Removes all unused mods and inactive lines from specified files in the PL. This prepares a PL for release or distribution outside of a development group. This command is restricted to the owner of the PL.

The `release` subcommand goes through each file processed, removes support for all versions except for the most recent one, removes all unused mods from the mods directory, removes all inactive UPDATE lines from the named files, and removes the delta commentary up to this point.

`-p` Purges inactive modules from the PL history.

`-r` *VID* Removes information older than the specified version.

pl (Batch mode only) The PL to which the subcommand applies.

files The individual files to which the subcommand applies. If *files* is a directory, each file in that directory is processed. If *files* is a `.` symbol, each file in the PL is processed. If *files* is a `-` symbol, the list of files is read from standard input.

`set` [*option*] [*value*]

(Interactive mode only) Sets the value of `SMPREFIX` or the PL on which you are working.

Possible values for *option* are `SMPREFIX` and `pl`. If *option* is not specified, `sm` prompts for the value to change.

`sm unget` [`-A` *attribute*] [`-B`] [`-n`] [`-N` *named_branch*] [`-o`] [`-r` *VID*] [`-z`] *pl files*

Unlocks a version of the specified files from the PL. In interactive mode, *pl* is not specified.

`-A` *attribute* Unlocks the VID that has the specified attribute defined.

`-B` Uses the most recent VID on a sideline branch when computing VIDs.

`-n` Specifies retention of the edited file. By default, these files are removed upon successful unlocking of the module.

`-N` *named_branch*
Specifies that the VID of the module to be unlocked is the current VID of the specified sideline branch.

`-o` Forces unlocking of the version of the modules locked, regardless of the owner of the lock. Use of this option is restricted to the owner of the PL.

`-r` *VID* Specifies the VID to which the command applies.

`-s` Suppresses messages produced by this command.

`-z` Removes the unlocked files from the directory specified by the *files* argument. By default, unlocked files are removed from the current directory.

pl (Batch mode only) The PL to which the command applies.

files The files to which the command refers in the PL. If *files* is a directory, each file in the directory is retrieved. If *files* is a `.` symbol, all files maintained in the PL are retrieved. If *files* is a `-`, the list of files is read from standard input.

`sm validate` [`-e` *string*] *pl*

Checks the validity of the specified PL. In interactive mode, *pl* is not specified.

`-e` *string* Ensures that the specified string is found within all modules in the PL, in addition to the string that is enforced by the `admin` subcommand.

pl (Batch mode only) The PL to which the command applies.

!*shell_command*

(Interactive mode only) Escapes to the shell to execute a shell command.

EXAMPLES

Example 1: The following is an example of the interactive use of `sm`:

```
$ sm
sm:  SMPREFIX not set, where are your pls?
/usr/src
sm:  What is the PL you wish to work on?
prod/prog
sm:  /usr/src/prod/prog is the PL being worked on
```

Example 2: The following sequence of commands gets a list of files maintained in the PL (with `list`), extracts a file with intent to change (with `get`), changes the file (with the shell escape `!`), and then checks the changes back into the system (with `mod` and `add`):

```
SM1-> list
file1.f
dir/file2.f
file3.f
file4.f
SM2-> get -e file1.f
file1.f 1.0
next delta 1.1
SM3-> !vi file1.f          (Make changes with the editor)
SM4-> mod -m mod1 -k 'major bugfix' file1.f
comments?  (enter ^D when finished.)-----------------------
This change is an example.
^D
mod working
mod: mod1 created
SM5-> !vi mod1             (Look at mod)
SM6-> add mod1
add:  file1.f 1.1 released
changes successfully applied to prod/prog
SM7-> exit
$
```

Example 3: The following is an example of the batch (command-line) use of `sm`:

```
$ setenv SMPREFIX /usr/src
$ sm list prod/prog
file1.f
dir/file2.f
file3.f
file4.f
$ sm get -e prod/prog file1.f
file1.f 1.0
next delta 1.1
$ vi file1.f          (Make your changes)
$ sm mod -m mod1 -k 'major bugfix' prod/prog file1.f
comments?  (enter ^D when finished.)-----------------------
This change is an example.
^D
mod working
mod: mod1 created
$ vi mod1             (Look at mod)
$ sm add prod/prog mod1
add:  file1.f 1.1 released
changes successfully applied to prod/prog
$
```

SEE ALSO

`nupdate`(1)

UNICOS Source Manager (USM) User's Guide, Cray Research publication SG–2097

NAME

snmpget – Communicates with a network entity by using SNMP GET requests

SYNOPSIS

snmpget [-d] [-n *varnumber*] *host community* [-P *portnumber*] *variable-name* [*variable-name*]...

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The snmpget utility is a simple network management protocol (SNMP) application that uses the GET request to query for information on a network entity. You can specify one or more fully qualified object identifiers as arguments on the command line.

The snmpget utility accepts the following arguments:

-d	Directs the application to dump input and output packets.
-n *varnumber*	Specifies the number of variables requested per packet. This argument is useful for debugging the agent.
host	Specifies either a host name or an Internet address in dot notation.
community	Specifies the community name for the transaction with the remote system.
-P *portnumber*	Specifies an alternate port number at which to send requests to an agent.
variable-name	Specifies the fully qualified object identifier to be retrieved by the snmpget request.

EXAMPLES

The following command retrieves the sysDescr.0 and sysUpTime.0 variables:

```
snmpget  dang.cray.com  criccn  mgmt.mib.system.sysDescr.0 \
mgmt.mib.system.sysUpTime.0
```

The output is as follows:

```
Name: mgmt.mib.system.sysDescr.0
OCTET STRING- (ascii):   Kinetics FastPath2

Name: mgmt.mib.system.sysUpTime.0
Timeticks: (2270351) 6:18:23
```

If the network entity encounters an error while the request packet is being processed, an error packet is returned and a message is shown, which helps to determine the error in the request. If other variables were in the request, the request is resent without the variable in error.

SEE ALSO

RFC 1155, RFC 1157, RFC 1213

NAME

snmpgetnxt – Communicates with a network entity by using SNMP requests

SYNOPSIS

snmpgetnxt [-d] *host community* [-P *portnumber*] *variable-name* [*variable-name*]...

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The snmpgetnxt utility is a simple network management protocol (SNMP) application that uses the GET NEXT request to query for information on a network entity. You can specify one or more object identifiers as arguments on the command line. For each one, the variable that is lexicographically "next" in the remote entity's management information base (MIB) is returned.

The snmpgetnxt utility accepts the following arguments:

-d	Directs the application to dump input and output packets.
host	Specifies either a host name or an Internet address in dot notation.
community	Specifies the community name for the transaction with the remote system.
-P *portnumber*	Sends requests on port *portnumber*. You must configure the snmpd daemon to listen on this port rather than on the default port 161. This option is used for debugging.
variable-name	Specifies the fully qualified object identifier to be retrieved by the snmpgetnxt request.

EXAMPLES

The following command retrieves the sysDescr.0 and sysUpTime.0 variables:

```
snmpgetnxt  dang.cray.com  criccn  mgmt.mib.system.sysDescr.0 \
mgmt.mib.system.sysUpTime.0
```

The output is as follows:

```
Name: mgmt.mib.system.sysObjectID.0
OBJECT IDENTIFIER:   .iso.org.dod.internet.private.enterprises.cray

Name: mgmt.mib.system.sysContact.0
OCTET STRING- (ascii): John Doe doe@cray.com
```

If the network entity encounters an error while processing the request packet, an error message is shown, which helps to determine the error in the request.

SEE ALSO

RFC 1155, RFC 1157, RFC 1213

NAME

snmpnetstat – Shows network status by using SNMP

SYNOPSIS

```
snmpnetstat host community [-P portnumber]
snmpnetstat host community [-an] [-P portnumber]
snmpnetstat host community [-inrs] [-P portnumber]
snmpnetstat host community [-P portnumber] [-n] [-I interface] interval
snmpnetstat host community [-P portnumber] [-p protocol]
```

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The snmpnetstat utility symbolically displays the values of various network-related information retrieved from a remote system by using the simple network management protocol (SNMP). There are several output formats, depending on the options for the information presented. The first form of the utility displays a list of active sockets. The second form presents the values of other network-related information according to the option selected. Using the third form, with an *interval* specified, snmpnetstat continuously displays the information about packet traffic on the configured network interfaces. The fourth form displays statistics about the specified protocol.

The snmpnetstat utility accepts the following arguments:

host	Specifies either a host name or an Internet address in dot notation.
community	Specifies the community name for the transaction with the remote system.
-P *portnumber*	Sends requests on port *portnumber*. You must configure the snmpd daemon to listen on this port rather than on the default port 161. This option is used for debugging.
-a	With the default display, shows the state of all sockets; usually sockets used by server processes are not shown.
-n	Shows network addresses as numbers (usually snmpnetstat interprets addresses and attempts to display them symbolically). You can use this option with any of the display formats.
-i	Shows the state of all interfaces.
-r	Shows the routing tables. When -s is also present, shows routing statistics instead.
-s	Shows per-protocol statistics.
-I *interface*	Shows information about only the specified interface; used with the *interval* argument.
interval	Specifies interval (in seconds) through which packet traffic information is displayed.

`-p` *protocol* Shows statistics about *protocol*, which is either a well-known name for a protocol or an alias for it. Some protocol names and aliases are listed in the `/etc/protocols` file. A null response typically means that there are no interesting numbers to report. If *protocol* is unknown or if no statistics routine for it exists, the program issues a warning.

For active sockets, the default display shows the local and remote addresses, protocol, and internal state of the protocol. If a socket's address specifies a network but no specific host address, address formats are of the form `host.port` or `network.port`. When known, the host and network addresses are displayed symbolically according to the `/etc/hosts` and `/etc/networks` databases, respectively. If a symbolic name for an address is unknown, or if the `-n` option is specified, the address is printed numerically, according to the address family. For more information about the Internet dot format, see `inet`(3C). Unspecified or wildcard addresses and ports appear as `*`.

The interface display provides a table of cumulative statistics about packets transferred, errors, and collisions. The network addresses of the interface and the maximum transmission unit (MTU) are also displayed.

The routing table display indicates the available routes and their status. Each route consists of a destination host or network and a gateway to use when forwarding packets. The *flags* field shows the state of the route (`U` if up), whether the route is to a gateway (`G`), whether the route was created dynamically by a redirect (`D`), and whether the route was modified by a redirect (`M`). Direct routes are created for each interface attached to the local host; the gateway field for such entries shows the address of the outgoing interface. The interface entry indicates the network interface used for the route.

When you invoke `snmpnetstat` with an *interval* argument, it displays a running count of statistics related to network interfaces. This display consists of a column for the primary interface and a column summarizing information for all interfaces. Use the `-I` option to replace the primary interface with another interface. The first line of each screen of information contains a summary since the system was last rebooted. Subsequent lines of output show values accumulated over the preceding interval.

EXAMPLES

The following `snmpnetstat` commands produce network statistics:

```
snq1-% snmpnetstat localhost criccn -i
Name      Mtu   Network     Address            Ipkts Ierrs    Opkts Oerrs
hy0*      16432 none        none                   0     0        0     0
hy1       16432 none        none              112544     0    87800     0
vme0*     16432 none        none                   0     0        0     0
vme1*     16432 none        none                   0     0        0     0
lsx0*     16432 none        none                   0     0        0     0
hi0*      65528 none        none                   0     0        0     0
hi1*      65528 none        none                   0     0        0     0
unet0*    32880 none        none                   0     0        0     0
lo0       65535 none        none               49528     0    49534     0

snq1-% snmpnetstat localhost criccn -I hy1
Name        Mtu   Network     Address            Ipkts Ierrs    Opkts Oerrs
hy1         16432 none        none              113178     0    88523     0

snq1-% snmpnetstat localhost criccn
Active Internet Connections
Proto Recv-Q Send-Q  Local Address          Foreign Address        (state)
tcp        0      0  *.1272                *.*                    CLOSED
tcp        0      0  localhost.cray.c.1272  localhost.cray.c.sunrp TIMEWAIT
tcp        0      0  snq1.cray.com.telnet   berserkly.cray.c.1518  ESTABLISHED
tcp        0      0  snq1.cray.com.telnet   cherry28.cray.co.1934  TIMEWAIT
tcp        0      0  snq1.cray.com.telnet   fir21.cray.com.1083    ESTABLISHED
tcp        0      0  snq1.cray.com.telnet   palm15.cray.com.1093   ESTABLISHED
tcp        0      0  snq1.cray.com.telnet   palm15.cray.com.1094   ESTABLISHED
tcp        0      0  snq1.cray.com.telnet   sumac15.cray.com.1256  ESTABLISHED
tcp        0      0  snq1.cray.com.telnet   sumac15.cray.com.1257  ESTABLISHED
tcp        0      0  snq1.cray.com.telnet   sumac15.cray.com.1258  ESTABLISHED
tcp        0      0  snq1.cray.com.telnet   hose.cray.com.2946     ESTABLISHED
tcp        0      0  snq1.cray.com.login    palm03.cray.com.1021   ESTABLISHED
tcp        0      0  snq1.cray.com.login    palm10.cray.com.1022   ESTABLISHED
tcp        0      0  snq1.cray.com.login    poplar17.cray.co.1021  ESTABLISHED
tcp        0      0  snq1.cray.com.809      aspen18.cray.com.980   TIMEWAIT
tcp        0      0  snq1.cray.com.815      cherry28.cray.co.shell TIMEWAIT
```

SEE ALSO

`inet`(3C) in the *UNICOS System Libraries Reference Manual*, Cray Research publication SR–2080

`hosts`(5), `networks`(5), `protocols`(5), `services`(5) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

RFC 1157

NAME

`snmpstatus` – Retrieves important information from a network entity by using SNMP requests

SYNOPSIS

`snmpstatus` [`-d`] [`-P` *portnumber*] *host community*

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `snmpstatus` utility is a simple network management protocol (SNMP) application that retrieves several important statistics from a network entity.

The `snmpstatus` utility accepts the following arguments:

`-d`	Directs the application to dump input and output packets.
`-P` *portnumber*	Sends requests on port *portnumber*. You must configure the `snmpd` daemon to listen on this port, rather than on the default port 161. This option is used for debugging.
host	Specifies either a host name or an Internet address in dot notation.
community	Specifies the community name for the transaction with the remote system. If you do not specify this argument, the community name defaults to `public`.

The information returned is as follows:

- The IP address of the entity
- A textual description of the entity (`sysDescr.0`)
- The uptime of the entity (`sysUpTime.0`)
- The sum of received packets on all interfaces (`ifInUCastPkts.* + ifInNUCastPkts.*`)
- The sum of transmitted packets on all interfaces (`ifOutUCastPkts.* + ifOutNUCastPkts.*`)
- The number of IP input packets (`ipInReceives.0`)
- The number of IP output packets (`ipOutRequests.0`)

EXAMPLES

Example 1: The following `snmpstatus` utility produces statistical information:

```
snmpstatus netdev-kbox.cc.cmu.edu public
```

The output is as follows:

```
[128.2.56.220]=>[Kinetics FastPath2] Up: 1 day, 4:43:31
IP recv/trans packets 262874/39867 |
IP recv/trans packets 31603/15805
```

Example 2: The `snmpstatus` utility also checks the operational status of all interfaces (`ifOperStatus.*`); if it finds any that are not running, it reports the interfaces as in the following example:

```
2 interfaces are down!
```

If the network entity encounters an error while processing the request packet, an error packet is returned and a message is shown, which helps to determine the error in the request. `snmpstatus` attempts to reform its request to eliminate the malformed variable, but this variable will then be missing from the displayed data.

SEE ALSO

RFC 1155, RFC 1157, RFC 1213

NAME

`snmptest` – Communicates with a network entity by using SNMP requests

SYNOPSIS

`snmptest` [`-d`] [`-P` *portnumber*] *host community*

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `snmptest` utility is a flexible simple network management protocol (SNMP) application that can monitor and manage information on a network entity.

The `snmptest` utility accepts the following arguments:

`-d`	Directs the application to dump input and output packets.
`-P` *portnumber*	Sends requests on port *portnumber*. You must configure the `snmpd` daemon to listen on this port, rather than on the default port 161. This option is used for debugging.
host	Specifies either a host name or an Internet address in dot notation.
community	Specifies the community name for the transaction with the remote system.

After invoking the program, a command-line interpreter begins to accept commands. It prompts with the following request:

```
Please enter the variable name:
```

You can enter one or more variable names, one per line. A blank line is a command to send a request for each of the variables (in one packet) to the remote entity.

EXAMPLES

In the following `snmptest` utility, the `system.sysDescr.0` name is entered at the prompt:

```
snmptest netdev-kbox.cc.cmu.edu public
Please enter the variable name:  mgmt.mib.system.sysDescr.0
Please enter the variable name:
```

The following information about the request and reply packets is returned:

```
Name: system.sysDescr.0
OCTET STRING- (ascii):
```

On startup, the program defaults to sending a GET request packet. This can be changed to a GET NEXT request or a SET request by entering the $N or $S command, respectively. Entering $G returns you to the GET request mode.

The $D command toggles the dumping of each sent and received packet.

When in SET request mode, the prompt requests more information for each variable. The following prompt requests that you enter the type of the variable:

```
Please enter variable type [i|s|x|d|n|o|t|a]:

Type "i" for an integer, "s" for an octet string in ascii, "x" for an
octet string as hex bytes separated by whitespace, "d" for an octet
string as decimal bytes separated by whitespace, "a" for an ip address
in dotted IP notation, and "o" for an object identifier.
```

You are then prompted for a value, as follows:

```
Please enter new value:
```

If it is an integer value, enter the integer (in decimal). If it is a string, enter decimal numbers separated by white space, one per byte of the string. Again, to send the packet, enter a blank line at the prompt for the variable name.

To quit the program, enter $Q at the prompt.

SEE ALSO

RFC 1155, RFC 1157, RFC 1213

NAME

snmptrap – Sends an SNMP TRAP message to a host

SYNOPSIS

snmptrap [-P *portnumber*] *host community trap-type specific-type device-description* [-a *agent-addr*] [-d]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The snmptrap utility is a simple network management protocol (SNMP) application that forms and sends an SNMP TRAP message to a host.

The snmptrap utility accepts the following arguments:

-P *portnumber* Sends requests on port *portnumber*. You must configure the snmpd daemon to listen on this port rather than on the default port 161. This option is used for debugging.

host Specifies either a host name or an Internet address in dot notation.

community Specifies the community name for the transaction with the remote system.

trap-type Specifies the type of TRAP message being sent. Trap types are integers, defined as follows:

- 0 (Cold start) The sending protocol entity is reinitializing itself such that the agent's configuration or the protocol entity implementation can be altered.
- 1 (Warm start) The sending protocol entity is reinitializing itself such that neither the agent configuration nor the protocol entity implementation is altered.
- 2 (Link down) The sending protocol entity recognizes a failure in one of the communication links represented in the agent's configuration.
- 3 (Link up) The sending protocol entity recognizes that one of the communication links represented in the agent's configuration has come up.
- 4 (Authentication failure) The sending protocol entity is the addressee of a protocol message that is not properly authenticated. While implementations of the SNMP must be able to generate this trap, they must also be able to suppress the emission of such traps through an implementation-specific mechanism.
- 6 (Enterprise specific) The sending protocol entity recognizes that some enterprise-specific event has occurred.

specific-type Identifies the particular trap that occurred.

device-description
Provides a textual description of the device sending this trap, which is used as the value of a `system.sysDescr.0` variable sent in the variable list of this trap message.

`-a` *agent-addr* Changes the address from which the trap reports it is being sent; otherwise, the sending host's address is used. This argument is optional.

`-d` Directs the application to dump the input and output packets.

EXAMPLES

The following `snmptrap` utility sends a cold start trap to the specified machine:

```
snmptrap nic.andrew.cmu.edu public 0 0
'SUN 3/60: SUNOS4.0'
```

SEE ALSO

RFC 1155, RFC 1157, RFC 1213

NAME

snmptrapd – Receives and logs SNMP TRAP messages

SYNOPSIS

snmptrapd [-p] [-d] [-P *portnumber*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The snmptrapd utility is a simple network management protocol (SNMP) application that receives and logs SNMP TRAP messages sent to the SNMP-TRAP port (162) on the local machine.

The snmptrapd utility accepts the following options:

-p Prints trap messages to the standard output; otherwise, it uses syslogd(8) to log messages. These syslog messages are sent with the level of LOG_WARNING and, if available (usually on 4.3BSD systems), they are sent to the LOG_LOCAL0 facility.

Following is an example of a log message:

```
Sep 17 22:39:52 suffern snmptrapd: 128.2.13.41: Cold Start  \
       Trap (0) Uptime: 8 days, 0:35:46
```

-d Directs the application to dump input and output packets.

-P *portnumber* Sends requests on port *portnumber*. You must configure the snmpd daemon to listen on this port rather than on the default port 161. This option is used for debugging.

The snmptrapd utility must be run as root so that UDP port 162 can be opened.

EXAMPLES

The following is an example of the use of snmptrapd. The snmpd daemon sends the coldstart trap (last line of the example) when it is started.

```
# snmptrapd -p &
# sdaemon -k snmpd
Stopping daemon: snmpd.
# sdaemon -s snmpd
Starting daemon: snmpd.
# 128.162.82.6: Cold Start Trap (0) Uptime: 0:00:00
```

SEE ALSO

`syslogd`(8) in the *UNICOS Administrator Commands Reference Manual*, Cray Research publication SR–2022

RFC 1155, RFC 1157, RFC 1213

NAME

snmpwalk, snmpwalka – Communicates with a network entity by using SNMP requests

SYNOPSIS

snmpwalk *host community* [*variable-name*] [-d] [-P *portnumber*]

snmpwalka *host community* [*variable-name*] [-d] [-P *portnumber*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The snmpwalk command is a simple network management protocol (SNMP) application that uses GET NEXT requests to query for a tree of information about a network entity. snmpwalka performs the same function asynchronously; it does not wait for a response from the agent before issuing another request.

The snmpwalk and snmpwalka commands accept the following arguments:

host Specifies either a host name or an Internet address in dot notation.

community Specifies the community name for the transaction with the remote system.

variable-name Specifies the portion of the object identifier space that is searched, using GET NEXT requests. All variables in the subtree below the given variable are queried and their values presented to the user.

If the *variable-name* argument is not present, snmpwalk searches the whole Internet management information base (MIB).

-d Directs the application to dump input and output packets.

-P *portnumber* Sends requests on port *portnumber*. You must configure the snmpd daemon to listen on this port rather than on the default port 161. This option is used for debugging.

EXAMPLES

The following example retrieves the mgmt.mib system variables:

```
snmpwalk   netdev-kbox.cc.cmu.edu public mgmt.mib.system
```

The output is as follows:

```
Name:  system.sysDescr.0
OCTET  STRING- (ascii):   Kinetics FastPath2

Name:      system.sysObjectID.0
OBJECT      IDENTIFIER: .iso.org.dod.internet.private.enterprises.\
           CMU.sysID.CMU-KIP

Name: system.sysUpTime.0
Timeticks: (2291082) 6:21:50
```

If the network entity encounters an error while the request packet is being processed, an error packet is returned and a message is shown, which helps to determine the error in the request.

If the tree search causes attempts to search beyond the end of the MIB, the following message is displayed:

```
End of MIB.
```

SEE ALSO

RFC 1155, RFC 1157, RFC 1213

NAME

soelim – Resolves and eliminates .so requests from nroff(1) or troff(1) input

SYNOPSIS

soelim [*filename* ...]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The soelim command reads the specified files or the standard input and performs the textual inclusion implied by the nroff(1) directives of the form .so *somefile* when they appear at the beginning of input lines. This allows the placement of individual tables in separate files to be run as a part of a large document. This is useful since commands such as tbl(1) do not normally do this.

An argument consisting of a dash (-) is taken to be a file name corresponding to the standard input.

To suppress inclusion use a ´ symbol instead of a . symbol, as shown in the following example:

```
´ so /usr/ucblib/doctools/tmac/tmac.s
```

EXAMPLES

A sample usage of soelim follows:

```
soelim exum?.n | tbl | nroff -ms | col | lpr
```

SEE ALSO

nroff(1), tbl(1), troff(1)

NAME

sort – Sorts, merges, or sequence check text files

SYNOPSIS

sort [-m] [-o *output*] [-b] [-d] [-f] [-i] [-k *keydef*]... [-M] [-n] [-r] [-t *char*] [-T *dir*] [-u] [-y *kmem*] [-Y] [-z *recsz*] [*files*]

sort -c [-b] [-d] [-f] [-i] [-k *keydef*]... [-M] [-n] [-r] [-t *char*] [-T *dir*] [-u] [-y *kmem*] [-Y] [-z *recsz*] [*files*]

Obsolescent version: may not be supported in future releases:

sort [-m] [-u] [-o *output*] [-b] [-d] [-f] [-i] [-M] [-n] [-r] [-t *char*] [-T *dir*] [-y *kmem*] [-z *recsz*] [+*pos1*[-*pos2*]]... [*files*]

sort -c [-u] [-o *output*] [-b] [-d] [-f] [-i] [-M] [-n] [-r] [-t *char*] [-T *dir*] [-y *kmem*] [-z *recsz*] [+*pos1*[-*pos2*]]... [*files*]

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4
CRI extensions (-Y option)
AT&T extensions (-M, -T, and -y options)

DESCRIPTION

The sort utility performs the following functions:

- Sorts lines of all the specified files together and writes the result to the specified output.
- Merges lines of all the named (presorted) files together and writes the result to the specified output.
- Checks that a single input file is correctly presorted.

The standard input is read if you use the - as a file name or if no input files are named.

Comparisons are based on one or more sort keys extracted from each line of input. By default, there is one sort key, the entire input line, and ordering is lexicographic by bytes in machine-collating sequence.

The sort utility accepts the following options:

-c Checks that the input file is sorted according to the ordering rules; gives no output and affects only the exit code.

-m Merges only; the input files are already sorted.

-o *output* Specifies the name of an output file to use rather than the standard output. This file may be the same as one of the input files.

`-T` *dir* Specifies the name of a directory (*dir*) in which temporary files are made. If set, this overrides the `TMPDIR` environment variable. If neither the `-T` is specified nor `TMPDIR` is in the environment, temporary files are made in `P_tmpdir`, as defined in the `stdio.h` file.

`-u` Unique; suppresses all but one key in each set of lines having equal keys.

`-y` *kmem* Specifies the amount of memory (*kmem*) to be used for the sort. The amount of memory used by the sort has a large impact on its performance. Sorting a small file in a large amount of memory is a waste. If you omit this option, `sort` begins using a system default memory size, and it continues to use more as needed. *kmem* specifies the number of kilobytes of memory. If the administrative minimum or maximum is violated, the corresponding extremum will be used. Thus, `-y 0` starts with minimum memory.

`-Y` Uses the maximum amount of memory available to perform sorting.

`-z` *recsz* Records the size of the longest line read in the sort phase so that buffers can be allocated during the merge phase. If you omit the sort phase by using the `-c` or `-m` option, a popular system default size will be used. Lines longer than the buffer size causes `sort` to terminate abnormally. To prevent abnormal termination, supply the actual number of bytes in the longest line to be merged (or some larger value).

The following options override the default ordering rules. When ordering options appear independent of any key field specifications, the requested field ordering rules apply globally to all sort keys. When attached to a specific key (see `-k`), the specified ordering options override all global ordering options for that key. In the obsolescent forms, if one of more of these options follows a `+`*pos1* option, it affects only the key field specified by that preceding option.

`-d` Places in dictionary order; only letters, digits, `<space>`s, and `<tab>`s are significant in comparisons.

`-f` Considers all lowercase characters that have uppercase equivalents to be the uppercase equivalent for the purposes of comparison.

`-i` Ignores all characters that are nonprintable.

`-M` Compares as months. The first three non-`<blank>` characters of the field are folded to uppercase and compared so that `JAN` < `FEB` < ... < `DEC`. Fields that are not valid compare low to `JAN`. The `-M` option implies the `-b` option (see the following).

`-n` Restricts the sort key to an initial numeric string, consisting of optional `<blank>`s, optional minus sign, and zero or more digits with optional decimal point, which is sorted by arithmetic value. An empty digit string is treated as 0. Leading zeros and signs on zeros do not affect ordering. The `-n` option implies the `-b` option. The `-b` option is effective only when restricted sort key specifications are in effect.

`-r` Reverses the sense of comparisons.

To alter the treatment of field separators, use the following options:

`-b` Ignores leading <blank>s when determining the starting and ending positions of a restricted sort key. If you specify the `-b` option before the first `-k` option, it is applied to all `-k` options. Otherwise, the `-b` option can be attached independently to each `-k` *field_start* or *field_end* argument.

`-t` *char* Uses *char* as the field separator character; *char* is not considered to be part of a field (although it may be included in a sort key). Each occurrence of *char* is significant (for example, <*char*><*char*> delimits an empty field). If you omit the `-t` option, <blank> characters are used as default field separators; each maximal nonempty sequence of <blank> characters that follows a non-<blank> character is a field separator.

To specify sort keys, use the following options:

`-k` *keydef...* The *keydef* argument is a restricted sort key field definition. The format of this definition is as follows:

field_start[*type*][,*field_end*[*type*]]

The *field_start* and *field_end* arguments define a key field restricted to a portion of the line (see the NOTES section), and *type* is a modifier from the list of characters `b`, `d`, `f`, `i`, `M`, `n`, `r`. The `b` modifier behaves like the `-b` option, but it applies only to the *field_start* or *field_end* to which it is attached. The other modifiers behave like the corresponding options, but they apply only to the key field to which they are attached; they have this effect if specified with *field_start*, *field_end*, or both. If any modifier is attached to a *field_start* or to a *field_end*, no option applies to either. Multiple `-k` options are permitted and are significant in command line order.

When multiple key fields exist, later keys are compared only after all earlier keys compare equal. When you specify the `-u` option, lines that otherwise compare equal are ordered as if none of the options `-d`, `-f`, `-i`, `-M`, `-n`, or `-k` were present (but with `-r` still in effect, if it was specified) and with all bytes in the lines significant to the comparison.

+*pos1* (Obsolescent) Specifies the start position of a key field. (See the NOTES section.)

-*pos2* (Obsolescent) Specifies the end position of a key field. (See the NOTES section.)

NOTES

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
`system`, `secadm`	In a privileged administrator shell environment, shell-redirected I/O is not subject to file protections.
`sysadm`	Shell-redirected output is subject to security label restrictions.

If the PRIV_SU configuration option is enabled, shell-redirected I/O on behalf of the super user is not subject to file protections.

The following notation defines a key field that begins at *field_start* and ends at *field_end* inclusive, unless *field_start* falls beyond the end of the line or after *field_end*, in which case the key field is empty. A missing *field_end* means the last character of the line.

-k *field_start*[*type*][,*field_end*[*type*]]

A field comprises a maximal sequence of nonseparating characters and, if you omit -t, any preceding field separator.

The *field_start* portion of the *keydef* argument has the form:

field_number[.*first_character*]

Fields and characters within fields are numbered starting with 1. The *field_number* and *first_character* pieces are positive decimal integers and specify the first character to be used as part of a sort key. If .*first_character* is omitted, it refers to the first character of the field.

The *field_end* portion of the *keydef* argument has the form:

field_number[.*last_character*]

The *field_number* is as previously described for *field_start*. The *last_character* piece, interpreted as a nonnegative decimal integer, specifies the last character to be used as part of the sort key. If *last_character* evaluates to 0 or .*last_character* is omitted, it refers to the last character of the field specified by *field_number*.

If the -b option or b type modifier is in effect, characters within a field are counted from the first non-<blank> in the field. This applies separately to *first_character* and *last_character*.

The obsolescent [+*pos1* [-*pos2*]] options provide functionality equivalent to the -k *keydef* option. For comparison, the full formats of these options are as follows:

+*field0_number*[.*first0_character*][*type*] [-*field0_number*[.*first0_character*][*type*]]
-k *field_number*[.*first_character*][*type*][,*field_number*[.*last_character*][*type*]]

In the obsolescent form, fields (specified by *field0_number*) and characters within fields (specified by *first0_character*) are numbered from 0 instead of one. The optional type modifiers are the same in both forms. If .*first0_character* is omitted or *first0_character* evaluates to 0, it refers to the first character of the field. The -b option does not apply to -*pos2*.

The fully specified +*pos1* -*pos2* form with type modifiers *T* and *U*:

+*w.xT* -*y.zU*

is equivalent to the following:

undefined	(When *z* is 0, and *U* contains b and -t is present)
-k *w*+1.*x*+1*T*,*y*.0*U*	(When *z* is 0 otherwise)
-k *w*+1.*x*+1*T*,*y*+1.z*U*	(When *z* is greater than 0)

EXIT STATUS

The `sort` utility exits with one of the following values:

0 All input files were output successfully, or `-c` was specified and the input file was sorted correctly.

1 Under the `-c` option, the file was not ordered as specified, or if the `-c` and `-u` options were both specified, two input lines that have equal keys were found. This exit status is not returned if you omit the `-c` option.

>1 An error occurred.

MESSAGES

Comments and exits with nonzero status for various trouble conditions (such as when input lines are too long), and for disorder discovered under the `-c` option.

When the last line of an input file is missing a `<newline>` character, `sort` appends one, prints a warning message, and continues.

EXAMPLES

Each of the following examples shows two command lines. The first performs the sort using the obsolescent version of `sort`; the second performs the sort using the standard `sort`.

Example 1: The following sorts the contents of `infile` with the second field as the sort key:

```
sort +1 -2 infile
sort -k 2,2 infile
```

Example 2: The following sorts, in reverse order, the contents of `infile1` and `infile2`, placing the output in `outfile` and using the first character of the second field as the sort key:

```
sort -r -o outfile +1.0 -1.2 infile1 infile2
sort -r -o outfile -k 2.1,2.2 infile1 infile2
```

Example 3: The following sorts, in reverse order, the contents of `infile1` and `infile2`, using the first non-`<blank>` character of the second field as the sort key:

```
sort -r +1.0b -1.1b infile1 infile2
sort -r -b -k 2.1,2.1 infile1 infile2
```

Example 4: The following prints the password file (`passwd`(5)), sorted by the numeric user ID (the third colon-separated field):

```
sort -t: +2n -3 /etc/passwd
sort -t: -n -k 3,3 /etc/passwd
```

Example 5: The following prints the lines of the already sorted `infile` file, suppressing all but the first occurrence of lines having the same third field (the `-um` options with just one input file makes the choice of a unique representative from a set of equal lines predictable):

```
sort -um +2 -3 infile
sort -um -k 3,3 infile
```

FILES

`/usr/tmp/stm*`	Temporary working file(s)
`/usr/tmp` `/tmp`	Default temporary directories
`TMPDIR`	User's temporary directory

SEE ALSO

`comm`(1), `csort`(1), `join`(1), `uniq`(1)

NAME

spacl – Manages an access control list (ACL)

SYNOPSIS

```
spacl -a [-i modfile] [-l] aclfile
spacl -a [-i modfile] [-s] aclfile
spacl -i modfile [-l] aclfile
spacl -i modfile [-s] aclfile
spacl -l [-t tmodfile] aclfile
spacl -r [-i modfile] [-l] aclfile
spacl -r [-i modfile] [-s] aclfile
spacl -s [-t tmodfile] aclfile
spacl -t tmodfile aclfile
```

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The spacl utility creates and maintains an access control list (ACL) file. An ACL file contains data entries that define the allowed access to a file on a specific user and/or group basis. The permissions defined in the ACL are called the absolute permissions. The absolute permissions in the ACL entries are intersected with the file's mask bits to determine the type of access allowed; this is called the effective permissions.

An ACL file consists of multiple entries, one entry per user/group name pair. The following information must be supplied for each ACL entry:

user Defines the user's login name. You can specify a valid user name, a wildcard character (*), or a question mark (?). When used in an add entry, the * represents all users. In a remove entry, the * represents only those ACL entries that specified * for the user.

The ? is used in remove entries only, to remove all ACL entries for the specified group (?:*gid*:).

group You can specify a valid group name, a wildcard character (*), a question mark (?), or a blank ().

In an add entry, the * represents all groups. In a remove entry, the * represents only those ACL entries that specified * for the group.

The ? is used in remove entries only, to remove all ACL entries for the specified user (*uid*:?:).

You can use a blank () with the wildcard character (*) to represent the owning group (*::). An entry of *:?: removes all group-only entries and the owning group entry.

permissions Permissions for access. Permissions are specified as follows:

`r` Grants read permission

`w` Grants write permission

`x` Grants execute permission

`n` Denies access

Any combination of `r`, `w`, and `x`, or `n` can be specified.

Defining duplicate entries (same *uid*`:`*gid*`:` pair) is not allowed.

There are four different types of ACL entries. Note that the format of entries are shown in pseudo code for these definitions (in the form of *user:group*). They are as follows:

User-only The absolute permissions defined for a specific user, regardless of the user's current groups. The format for this type of entry is *uid*`:*:`.

User-group The absolute permissions defined for a specific user when that user is a member of a specific group. The format for this type of entry is *uid*`:`*gid*`:`.

Group-only The absolute permissions defined for any user that is a member of the specified group. The format for this type of entry is `*:`*gid*`:`.

Owning-group The absolute permissions defined for the group that owns the file. The format for this type of entry is `*::`.

The following is a description of the file parameter types:

aclfile The *aclfile* is the file that results from the execution of `spacl` by any user. If the file does not exist, `spacl` creates it. It is stored as a binary file. The `spacl` command fails when *aclfile* is not specified or when an invalid *aclfile* is specified (that is, when the file does not conform to an `acl.h` format). Use the `spset -a` command to apply an *aclfile* to a file.

modfile The user-generated file that contains add and/or remove statements. The following is the format for adding a record via *modfile*:

`a:` *user_name* `:` *group_name* `:` *access_mode* `:`

The following is the format for removing a record via *modfile*:

`r:` *user_name* `:` *group_name* `:`

tmodfile An ASCII text that is created by using the `spget -a` or `spacl -l` command and redirecting the output to *tmodfile*. This file may be modified by using a text editor and then used as input to create the binary *aclfile*.

`spacl` accepts the following options:

`-a` *aclfile* Interactively adds entries to an ACL file named *aclfile*. You are prompted for the user's name, group name, and the access modes. Enter `quit` for user, group, or access mode to exit the interactive session. Press `<CONTROL-c>` to discard all changes and exit the interactive session.

`-i` *modfile* Inputs ACL edit statements via *modfile*.

`-l` Lists the contents of the ACL file named *aclfile*. If both the `-a` and `-l` options are specified, the listed information is the contents of *aclfile* after exiting the interactive session.

`-r` *aclfile* Interactively removes entries from an ACL file named *aclfile*. You are prompted for the user's name, group name, and the access modes. Enter `quit` for user, group, or access mode to exit the interactive session. Press `<CONTROL-c>` to discard all changes and exit the interactive session.

`-s` *aclfile* Lists the contents of *aclfile* in a short format.

`-t` *tmodfile aclfile*
Creates *aclfile* by using *tmodfile* as input.

NOTES

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
`system`, `secadm`	In a privileged administrator shell environment, shell-redirected I/O is not subject to file protections.
`sysadm`	Shell-redirected output is subject to security label restrictions.

If the `PRIV_SU` configuration option is enabled, shell-redirected I/O on behalf of the super user is not subject to file protections.

The NFS protocol was not designed to handle ACL information. The `spset`(1) command cannot set ACLs on files that reside on NFS-mounted file systems. The `spget`(1) command can display an empty ACL for all files that reside on NFS-mounted file systems.

The completion of an `spacl` command produces an updated, sorted ACL binary file named by the *aclfile* operand.

The `-a` and `-r` options cannot be processed concurrently.

EXAMPLES

Example 1: The following example shows how to interactively add entries to the ACL file called `siteacl`. The first entry adds an entry that defines absolute read and write access for `beth`, regardless of her groups. The second entry defines no access for the owning group. User input is shown in bold:

```
$ spacl -a siteacl

ENTER "quit" to END SESSION, "ctrl-c" to ABORT

enter user’s name OR an * ..... beth
enter group name OR an * ..... *
enter access mode (n = none) ...rw

ADD MODE

enter user’s name OR an * ..... *
enter group name OR a blank ...
enter access mode (n = none) ...n

ADD MODE

enter user’s name OR an * ..... quit
entry session terminated
```

Example 2: The following example shows how to interactively remove entries from an ACL called `newacl`. The first entry removes the entry that defines both the user `henry` and the group `testing`. The second entry removes all group-only entries (including the owning-group entry). User input is shown in bold:

```
$ spacl -r newacl

ENTER "quit" to END SESSION, "ctrl-c" to ABORT

REMOVE MODE

enter user's name OR a ? OR an * ...... henry
enter group name OR a ? OR an * ....... testing

REMOVE MODE

enter user's name OR a ? OR an * ...... *
enter group name OR a ? OR an * ....... ?

REMOVE MODE

enter user's name OR a ? OR an * ...... quit
entry session terminated
```

Example 3: The following example shows a file that contains edit statements for an ACL. The first line adds an entry for `bill` that allows absolute read and execute access regardless of his groups. The second line removes the entry for the user `norma` and the group `testing`. The third line adds an entry that allows no access for the owning group. The fourth line removes all user-only entries. All lines must be terminated with a colon.

```
a : bill : * : rx :
r : norma : testing :
a : * :   : n :
r : ? : * :
```

Example 4: The following example shows how to apply the file generated in the previous example (called `changes`) to the ACL file called `newacl`:

```
$ spacl -i changes newacl
```

Example 5: The following example shows the command line that applies the add/remove edit statements in `changes` to `newacl`, and then enters an interactive entry session. Upon exiting the interactive session, the contents of `newacl` is displayed:

```
$ spacl -a -l -i changes newacl
```

Example 6: An *aclfile* can be created by redirecting a *tmodfile*, as shown in the following example. The first command line shows that `tmodfile` is created from an ACL file. The second command line shows `tmodfile` being created from the ACL controls set on the file:

```
$ spacl -l newacl > tmodfile

$ spget -a foo.c > tmodfile
```

Example 7: The following example shows how to create a new ACL file called `acltest` using the `tmodfile` created in the previous example:

```
$ spacl -t tmodfile acltest
```

Example 8: The following example shows the command line that creates an ACL file named `t3` from the modfile `tlo`, then displays the contents of the new `ACL`:

```
$ spacl -l -t tlo t3
```

In the previous example, the `-s` option could have been used in place of the `-l` option, but only the short form of the contents of the ACL would be displayed.

Example 9: The following example shows the command line that creates the ACL called `myacl`, then applies the changes specified in `change` to `myacl`. The interactive add mode is entered, where you can add entries. Upon completion, the entries in `myacl` are listed:

```
$ spacl -a -l -i change myacl
```

FILES

`/usr/include/sys/acl.h`

SEE ALSO

`spclr`(1), `spset`(1)

`getfacl`(2), `rmfacl`(2), `setfacl`(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

`acl`(5) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

General UNICOS System Administration, Cray Research publication SG–2301

NAME

spclr – Performs secure clear operations

SYNOPSIS

```
spclr [-m] [-i] [-r] files
spclr -d [-m] [-i] [-r] [-p] files
spclr -s [-m] [-i] [-r] files
spclr -a [-m] [-i] [-r] [-F] [-V] files
spclr -M [-K] [-a] [-m] [-i] segments
spclr -Q [-K] [-a] [-m] [-i] queues
spclr -S [-K] [-a] [-m] [-i] semaphores
```

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The spclr utility removes files and overwrites (clears) associated disk space with a pattern. The spclr utility performs disk space clearing and declassification procedures in accordance with Department of Defense guidelines. This utility also removes access control lists (ACLs) assigned to files, directories, or IPC objects (shared memory segments, message queues, and semaphores).

If spclr is called without the -d, -s or -a option, it functions similar to the rm(1) utility. In other words, the file is removed, but the disk space associated with the file is not overwritten.

The -F and -V options are available only if the Cray/REELlibrarian (CRL) has been installed on the UNICOS system.

The spclr utility accepts the following options:

-F Interprets *files* as a list of CRL files. The format of the elements of *files* is one of the following:

```
V:volume_set_name  file_name
V:volume_set_name S:file_sequence
file_name
```

The *volume_set_name* variable is an existing CRL volume set name or *.vsid*, where *vsid* is the first volume ID of a volume set. *file_sequence* is the numeric file sequence of the CRL file. *file_name* is the file ID of the CRL file. The V:*volume_set_name S*:*file_sequence* format is always unambiguous, whereas the other two formats are not.

-V Interprets *files* as a list of CRL volume sets. The format of the elements of *files* is one of the following:

```
volume_set_name
.vsid
```

The *volume_set_name* variable is an existing CRL volume set name. *vsid* is the first volume ID of a volume set.

`-M` *segments*
Interprets *segments* as a list of shared memory segments. By default, these are named using the shared memory identifiers displayed by the `ipcs`(1) command. See the `-K` option explanation.

`-Q` *queues*
Interprets *queues* as a list of message queues. By default, these are named using the message queue Identifiers displayed by the `ipcs`(1) command. See the `-K` option explanation.

`-S` *semaphores*
Interprets *semaphores* as a list of semaphores. By default, these are named using the semaphore identifiers displayed by the `ipcs`(1) command. See the `-K` option explanation.

`-K` Specifies that the message queue, shared memory, or semaphore names specified in the arguments are keys as displayed by `ipcs`(1), not identifiers.

`-d` Declassifies disk space associated with *files*. Space is overwritten with the `DECLASSIFY_PATTERN` configuration parameter, then with the negated pattern, then with the original pattern. This option is available only when the `DECLASSIFY_DISK` configuration parameter is enabled. It cannot be used with the `-a` or `-s` options.

`-p` Uses a random pattern when declassifying disk space associated with *files*. Space is overwritten with the `DECLASSIFY_PATTERN` configuration parameter, then with the negated pattern, then with a random pattern. This option is valid only with the `-d` option.

`-m` Disables printing of error messages.

`-i` Requests permission before removing a file or the file's ACL.

`-r` Recursively removes the contents of a directory, its subdirectories, and the directory itself. If the `-a` option is specified, only the ACLs of the contents of a directory, its subdirectories, and the directory itself are recursively removed.

`-s` Sanitizes disk space associated with *files*. Space is filled with a pattern that is determined by the `SANITIZE_PATTERN` configuration parameter. Cannot be used with `-d`, `-a`, or `-p` options.

`-a` Removes ACL associated with *files*. Cannot be used with `-d`, `-s`, or `-p` options.

The group access mode is set to the mode granted the owning group before the ACL is removed (that is, the mode granted when the ACL was set on the object).

The `setuid` and `setgid` bits are cleared if the caller is not appropriately authorized.

NOTES

When the path name supplied to the `spclr` utility specifies a multilevel symbolic link (the name of a multilevel directory), the attributes are changed only on the root of the multilevel directory tree. In the case of ACLs, this affects all subsequently created labeled subdirectories. In any case, the attribute change does not affect existing labeled subdirectories. To set attributes on the existing labeled subdirectories, you must specify the path names of the existing labeled subdirectory found in the root of the multilevel directory to the `spclr` utility.

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the action shown:

Active Category	Action
`system`, `secadm`	Allowed to perform all `spclr` operations on any file. For the `-a` option, the file `setuid` and `setgid` bits are not cleared.
`sysadm`	Allowed to perform all `spclr` operations on files, subject to security label restrictions. For the `-a` option, the file `setuid` and `setgid` bits are not cleared.

If `PRIV_SU` is enabled, the super user is allowed to perform all `spclr` operations on any file. The super user or a user with the `suidgid` permission can override the clearing of the file's `setuid` and `setgid` bits.

An appropriately authorized administrator can set the declassify pattern and number of overwrites during UNICOS system configuration. The administrator can also change the sanitize pattern at this time. See *General UNICOS System Administration*, Cray Research publication SG–2301.

SEE ALSO

`ipcs`(1), `rm`(1), `spacl`(1), `spset`(1)

`getsysv`(2), `rmfacl`(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

Cray/REELlibrarian (CRL) User's Guide, Cray Research publication SG–2126

Cray/REELlibrarian (CRL) Administrator's Guide, Cray Research publication SG–2127

General UNICOS System Administration, Cray Research publication SG–2301

Department of Defense Magnetic Remanence Security Guideline, CSC-STD-005-85, November 15, 1985

NAME

split – Splits files into pieces

SYNOPSIS

split [-l *line_count*] [-a *suffix_length*] [-v] [*file* [*name*]]
split -b *n*[*unit*] [-a *suffix_length*] [-v] [*file* [*name*]]

Obsolescent version: May not be supported in future releases:

split [-*line_count*] [-a *suffix_length*] [-v] [*file* [*name*]]

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The split utility reads *file* and writes it in *n*-line pieces (default 1000 lines) onto a set of output files (*name*). To modify the size of the output files, use the -b or -l option. Each output file is created with a unique suffix, which consists of exactly *suffix_length* lowercase letters from the POSIX local(2). By default, the names of the output files are x, followed by a two-character suffix, starting with aa, ab, ac, and so on, and continuing until the suffix zz, for a maximum of 676 files.

If the number of files required exceeds the maximum allowed by the suffix length provided, split fails after creating the last file with a valid suffix.

The split utility accepts the following options and operands:

-a *suffix_length*
: Use *suffix_length* letters to form the suffix part of the file names of the split file. If you omit -a, the default suffix length is 2. If the sum of the *name* operand and the *suffix_length* argument would create a file name that exceeds {NAME_MAX} bytes, an error occurs; split then exits and no files are created.

-b *n*[*unit*]
: Split a file into pieces *n* bytes in size. If a *unit* is k, the file is split into pieces *n**1024 bytes in size. If a *unit* is m, the file is split into pieces *n**1048576 bytes in size.

-l *line_count*
-*line_count* (Obsolescent)
: Indicates the number of lines in each piece. Default is 1000 lines. If the input does not end with a <newline>, the partial line is included in the last output file.

-v
: Writes out each output file name to standard output as it is created.

file Specifies the path name of the ordinary file to be split. If you do not specify an input file or *file* is `-`, the standard input is used.

name Specifies the prefix to be used for each of the files resulting from the split operation. If you omit the *name* argument, `x` is used as the prefix of the output files. The combined length of the basename of *prefix* and *suffix_length* cannot exceed {`NAME_MAX`} bytes.

NOTES

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
`system`, `secadm`	Allowed to manage any file. In a privileged administrator shell environment, shell-redirected I/O is not subject to file protections.
`sysadm`	Allowed to manage any file subject to security label restrictions. Shell-redirected I/O is subject to security label restrictions.

If the `PRIV_SU` configuration option is enabled, the super user is allowed to manage any file. Shell-redirected I/O on behalf of the super user is not subject to file protections.

EXIT STATUS

The `split` utility exits with one of the following values:

0 Successful completion.

>0 An error occurred.

SEE ALSO

`csplit`(1)

NAME

spset, spget – Sets and displays security attributes

SYNOPSIS

```
spset [-a aclfile] [-c cmp] [-d facl] [-f] [-i cls] [-j cat] [-k flgs] [-l lvl] [-F] [-V] files
spset -M [-K] [-a aclfile] [-c cmp] [-d facl] [-f] [-l lvl] segments
spset -Q [-K] [-a aclfile] [-c cmp] [-d facl] [-f] [-l lvl] queues
spset -S [-K] [-a aclfile] [-c cmp] [-d facl] [-f] [-l lvl] semaphores
spset -s min max cmps

spget [-a[r][-e]] [-f] [-F] [-V] files
spget -M [-K] [-a[r][-e]] [-f] segments
spget -Q [-K] [-a[r][-e]] [-f] queues
spget -S [-K] [-a[r][-e]] [-f] semaphores
spget [-s]
```

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The spset and spget utilities allow you to manipulate the security environment. Use spset to set attributes and spget to display those attributes. Attempts by unauthorized users to change security attributes can be audited.

The -F and -V options are available only if the Cray/REELlibrarian (CRL) has been installed on the UNICOS system.

The spset utility accepts the following options:

-a *aclfile* Assigns the access control list (ACL) to the indicated files and/or directories. *aclfile* is a file containing the ACL (created with spacl(1)) and *files* is one or more file and/or directory names to which the ACL is assigned. The -a option can fail for the following reasons:

- You are not the owner of the *files* or you are not appropriately privileged.
- Either the *files* or *aclfile* do not exist (are not valid).
- The *aclfile* file is not a legitimate ACL.

The group access mode is set to the mode granted the owning group before the ACL is removed (that is, the mode granted when the ACL was set on the object).

The setuid and setgid bits are cleared if the caller is not appropriately privileged.

Note that the `spset -a` utility does not result in any connection between the *aclfile* and the *files*. The data from the *aclfile* is converted to binary form and moved to a block named in the inode. Thus, if *aclfile* is modified or removed after `spset -a` is executed, the ACL permissions assigned to *files* are not changed. To change the ACL permissions, another ACL must be assigned to the file. To remove an ACL from a file, the `spclr`(1) utility must be used.

`-c` *cmp* Sets the file compartments (*cmp*) for one or more files. The requester must be an appropriately authorized user. *cmp* can be either the octal representation of specific compartments or a string of one or more (comma-separated) compartment names as defined in `uts/cf/seclabs.c`. The user must specify compartments within the authorized set for the file system on which the file resides.

If the `-V` option is used, *cmp* can be in the *lower_set upper_set* format if you are setting a multilevel CRL volume set. *upper_set* must dominate *lower_set*.

`-d` *facl* Assigns an ACL to the indicated files and/or directories by duplicating the ACL permissions assigned to *facl*. All rules pertaining to the `-a` option apply. If the `-M`, `-Q`, or `-S` option is specified, the *facl* argument is interpreted as a file name and not as an IPC object name.

`-f` Forces the assignment of a new ACL. If a file already has an ACL, `spset` asks whether the existing ACL should be replaced if the `-f` option is not selected. Using the `-f` option suppresses the question and replaces the ACL.

`-i` *cls* Sets the file integrity class for one or more files. The requester must be an an appropriately authorized user. This option is not supported.

`-j` *cat* Sets the file categories for one or more files. The requester must be an appropriately authorized user. This option is not supported.

`-k` *flgs* Sets the file security flags for one or more files. The requester must be an an appropriately authorized user. *flgs* can be either the octal representation of specific security flags or a string of one or more (comma-separated) flag names as defined in `/usr/include/sys/tfm.h`. The `exec` flag is obsolete. It is not possible to enable the secure device flag (`secdv`) using `spset`. To enable this flag, the `spdev`(8) command must be used.

`-l` *lvl* Sets the file security level (*lvl*) for one or more files. The requester must be an appropriately authorized user. The file's security level must fit within the boundaries established by the lower and upper security levels of the file system in which the file resides. *lvl* can be either a decimal integer or a mnemonic level name, as defined in `uts/cf/seclabs.c`.

If the `-V` option is used, and you are setting a multilevel CRL volume set, the format can be *lower-upper*.

`-M` *segments* Interprets *segments* as a list of shared memory segments. By default, these are named using the shared memory identifiers displayed by the `ipcs`(1) command. See the `-K` option explanation.

`-Q` *queues* Interprets *queues* as a list of message queues. By default, these are named using the message queue identifiers displayed by the `ipcs`(1) command. See the `-K` option explanation.

-S *semaphores*
Interprets *semaphores* as a list of semaphores. By default, these are named using the semaphore identifiers displayed by the ipcs(1) command. See the -K option explanation.

-K Specifies that the message queue, shared memory, or semaphore names specified in the arguments are keys as displayed by ipcs(1), not identifiers.

-F Interprets *files* as a list of CRL files. The format of the elements of *files* is one of the following:

V:*volume_set_name file_name*
V:*volume_set_name S*:*file_sequence*
file_name

The *volume_set_name* variable is an existing CRL volume set name or *.vsid*, where *vsid* is the first volume ID of a volume set. *file_sequence* is the numeric file sequence of the CRL file. *file_name* is the file ID of the CRL file. The V:*volume_set_name S*:*file_sequence* format is always unambiguous, whereas the other two format are not.

-V Interprets *files* as a list of CRL volume sets. The format of the elements of *files* is one of the following:

volume_set_name
.*vsid*

The *volume_set_name* variable is an existing CRL volume set name. *vsid* is the first volume ID of a volume.

files Specifies the file that contains the attributes that are being changed.

-s *min max cmps*
Sets the minimum and maximum system security level and the system's authorized compartments. The requester must be an an appropriately authorized user. *cmps* can be either the octal representation of specific compartments or a string of one or more (comma-separated) compartment names as defined in uts/cf/seclabs.c. In addition, the following rules must be satisfied:

- *min* must be less than or equal to the minimum security level for all mounted file systems, with the exception of file systems that are labeled with the syslow security label.
- *max* must be greater than or equal to the maximum security level for all mounted file systems, with the exception of file systems that are labeled with the syshigh security label.
- *cmps* must dominate all authorized compartments for all mounted file systems.

The spget utility accepts the following options:

No options Displays the user's security environment (permissions, security levels, compartments, integrity class, and categories). The active and maximum integrity class fields are not supported.

`-a` Displays the ACL information for the specified files.

`-r` Displays the ACL information in reduced format. Valid only with the `-a` option.

`-f` Displays the security attributes (security level, compartments, integrity class, categories, and flags) for the specified objects. If you specify both the `-a` and `-f` options, both the ACL and the security attribute information is displayed. Message queues, shared memory segments, and semaphores have only security levels and compartments, so only these security attributes are displayed. If you specify the `-M`, `-S`, or `-Q` option, but do not specify the `-a` option, specifying the `-f` option is not needed. The file category and class fields are not supported.

`-s` Displays the system security minimum and maximum security levels and authorized compartments.

`-e` Displays the masked ACL permissions. Valid only with the `-a` option. This option masks each ACL entry's mode against the file's permissions and displays the resultant mode.

`-M` *segments* Obtains the attributes of the shared memory segments named in *segments*. By default, these are named using the shared memory identifiers displayed by the `ipcs`(1) command. See the `-K` option explanation.

`-Q` *queues* Obtains the attributes of the message queues named in *queues*. By default, these are named using the message queue identifiers displayed by the `ipcs`(1) command. See the `-K` option explanation.

`-S` *semaphores*
Obtains the attributes of the semaphores named in *semaphores*. By default, these are named using the semaphore identifiers displayed by the `ipcs`(1) command. See the `-K` option explanation.

`-K` Specifies that the message queue, shared memory, or semaphore names specified in the arguments are keys as displayed by `ipcs`(1), not identifiers.

`-F` Interprets *files* as a list of CRL files. See the description in the `spset` options list.

`-V` Interprets *files* as a list of CRL volume sets. See the description in the `spset` options list.

Compartments, permissions, and categories are displayed in octal and by name. Levels and classes are displayed in decimal and by name. Your active security attributes are those at which you are currently operating.

NOTES

When the path name supplied to the `spset` utility specifies a multilevel symbolic link (the name of a multilevel directory), the attributes are changed only on the root of the multilevel directory tree. In the case of ACLs, this affects all subsequently created labeled subdirectories. In any case, the attribute change does not affect existing labeled subdirectories. To set attributes on the existing labeled subdirectories, you must specify the path names of the existing labeled subdirectory found in the root of the multilevel directory to the `spset` utility.

If the level and compartments are both to be set, the spdev(8) command must be used, since spset does not set level and compartments at the same time. Once the level or compartment has changed, the user loses write access to the directory and the other attributes can no longer be set.

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the action shown:

Active Category	Action
system, secadm	Allowed to perform all spset and spget operations on any file.
sysadm	Allowed to set and display file access control lists and display file security attributes, subject to security label restrictions. Allowed to retrieve system security attributes.

If the PRIV_SU configuration option is enabled, the super user is allowed to perform all spset and spget operations on any file.

EXIT STATUS

Returns 0 on successful completion; otherwise, returns nonzero.

EXAMPLES

Example 1: The following example sets the security level of the shared memory segments whose IPC keys are 0x12345 and 0x6789a to 5:

```
spset -l 5 -K -M 0x12345 0x6789a
```

Example 2: The following example displays the security labeling information on the shared memory segments whose IPC keys are 0x12345 and 0x6789a:

```
spget -K -M 0x12345 0x6789a
```

Example 3: The following example displays the ACL on the shared memory segments whose IPC keys are 0x12345 and 0x6789a:

```
spget -K -M -a 0x12345 0x6789a
```

Example 4: The following example sets the security level of the shared memory segments whose IPC identifiers are 1234 and 5678 to 5:

```
spset -l 5 -M 1234 5678
```

Example 5: The following example displays the security labeling information on the shared memory segments whose IPC identifiers are 1234 and 5678:

```
spget -M 1234 5678
```

Example 6: The following example displays the ACL on the shared memory segments whose IPC identifiers are `1234` and `5678`:

```
spget -M -a 1234 5678
```

Example 7: The following example displays the ACL and security labeling information on the shared memory segments whose IPC identifiers are `1234` and `5678`:

```
spget -M -a -f 1234 5678
```

Example 8: The following example displays both the ACL and the security labeling information on the shared memory segments whose IPC keys are `0x12345` and `0x6789a`:

```
spget -K -M -a -f 0x12345 0x6789a
```

You can substitute the `-S` or `-Q` options for the `-M` option in the previous examples to change from shared memory segments to semaphores or message queues information respectively, being set or displayed.

The previous examples use hexadecimal values for the keys and decimal values for the identifiers, because this is how they are typically displayed by the `ipcs`(1) command. Any decimal, octal, or hexadecimal value can be used for either the key or the identifier, as long it is specified in the standard form: 0xxxxx or 0Xxxxx for hexadecimal, 0xxxx for octal, and [1–9]xxxx for decimal.

SEE ALSO

`ipcs`(1), `setucmp`(1), `setulvl`(1), `spacl`(1), `spclr`(1)

`getfacl`(2), `getsysv`(2), `getusrv`(2), `secstat`(2), `setfacl`(2), `setfcmp`(2), `setfflg`(2), `setflvl`(2), `setsysv`(2), `setucat`(2), `setucmp`(2), `setulvl`(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

`spdev`(8) in the *UNICOS Administrator Commands Reference Manual*, Cray Research publication SR–2022

Cray/REELlibrarian (CRL) User's Guide, Cray Research publication SG–2126

Cray/REELlibrarian (CRL) Administrator's Guide, Cray Research publication SG–2127

General UNICOS System Administration, Cray Research publication SG–2301

NAME

strings – Finds printable strings in files

SYNOPSIS

strings [-a] [-f] [-n *number*] [-t *format*] [*files*]

Obsolescent version; may not be supported in future releases;

strings [-] [-f] [-o] [-t *format*] [-*number*] [*files*]

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4
BSD extensions (-f and -o options)

DESCRIPTION

The strings utility looks for printable strings in regular *files* and writes those strings to standard output. A character string is any sequence of 4 (the default) or more printable characters that end with a <newline> or a NULL character.

The strings utility accepts the following options:

-a
- (Obsolescent) Scans files in their entirety. If -a is not specified, the executable file starts after the scan of an exec(2) structure.

-f Precedes each string by the name of the file in which it was found.

-n *number*
-*number* (Obsolescent)
Specifies the minimum string length. *number* is a positive decimal integer. The default is 4.

-o
(Obsolescent) Causes each string to be preceded by its offset in the file (in octal). This is equivalent to -t o.

-t *format* Prints the offset in the *format* specified. *format* can be one of the following:

- d Decimal
- o Octal
- x Hexadecimal

files The path name of a regular file to be used as input. If no *file* operand is specified, input is read from the standard input.

The `strings` utility is useful for identifying random object files.

NOTES

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
`system`, `secadm`	Allowed to search any file. In a privileged administrator shell environment, shell-redirected I/O is not subject to file protections.
`sysadm`	Allowed to search any file subject to security label restrictions. Shell-redirected I/O is subject to security label restrictions.

If the `PRIV_SU` configuration option is enabled, the super user is allowed to search any file. Shell-redirected I/O on behalf of the super user is not subject to file protections.

EXIT STATUS

The `strings` utility exits with one of the following values:

0 Successful completion.

>0 An error occurred.

BUGS

The algorithm for identifying strings is extremely primitive. In particular, machine code instructions on certain architectures can resemble sequences of ASCII bytes, which will fool the algorithm.

SEE ALSO

`od`(1)

`exec`(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

NAME

`strip` – Removes symbol table from an executable file

SYNOPSIS

`strip` [`-s`] [`-V`] *files*

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4
CRI extensions (`-s` and `-V` options)

DESCRIPTION

The `strip` utility removes the symbol table information from executable files. After this has been done, no symbolic debugging access is available from files.

This is useful for saving disk space after a program has been debugged (there is no effect on memory used at run time by the program). The effect of `strip` is the same as using the `-s` option on `segldr`(1) or `ld`(1).

The `strip` utility accepts the following options:

`-s` Prints to standard output a summary of the files that have been successfully stripped. The summary includes the name of the file, the original size of the file in bytes, the size of the file in bytes with the symbol table removed, the size in bytes of the symbol table removed, and the percentage of the original file that consisted of symbol tables.

`-V` Outputs the `strip` version number to standard error.

files Specifies the files to be stripped.

NOTES

A stripped executable file should not be used for profiling with `prof`(1) and cannot be used with debuggers.

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
`system`, `secadm`	Allowed to remove the symbol table from any executable file. In a privileged administrator shell environment, shell-redirected I/O is not subject to file protections.
`sysadm`	Allowed to remove the symbol table from any executable file subject to security label restrictions. Shell-redirected I/O is subject to security label restrictions.

If the PRIV_SU configuration option is enabled, the super user is allowed to remove the symbol table from any executable file. Shell-redirected I/O on behalf of the super user is not subject to file protections.

EXIT STATUS

The exit status matches the number of files that failed to have their symbol tables removed.

FILES

a.out

SEE ALSO

ar(1)
ld(1) to invoke the link editor
prof(1) to show where execution time is spent
segldr(1) to invoke the CRI segment loader (SEGLDR)

a.out(5) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

NAME

stty – Sets the options for a terminal

SYNOPSIS

```
stty -a
stty -g
stty [options]
```

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The stty utility sets certain terminal I/O options for the device that is the current standard input; without arguments, it reports the settings of certain options.

In this report, if a character is preceded by a caret (^), the value of that option is the corresponding CONTROL character (for example, ^h is <CONTROL-h>; in this case, recall that <CONTROL-h> is the same as the backspace key.) The sequence ^´ or ^@ means that an option has a null value.

The stty utility accepts the following options:

-a Writes to standard output all of the current settings for the terminal.

-g Writes to standard output all of the current settings in a form that can be used as an argument to another stty utility. (See Examples 4 and 5 in the EXAMPLES section.)

For detailed information about the modes listed the Control Modes through Local Modes subsections, see termio(4). Options in the last group are implemented with options in the previous groups. Many combinations of options make no sense, but no sanity checking is performed. The options are selected from the following.

Control Modes

parenb (-parenb)	Enables (disables) parity generation and detection.
parext (-parext)	Enables (disables) extended parity generation and detection for mark and space parity.
parodd (-parodd)	Selects odd (even) parity.
cs5 cs6 cs7 cs8	Selects character size (see termio(4)).
0	Hangs up phone line immediately.

`110 300 600 1200 1800 2400 4800 9600 19200 38400`
Sets terminal baud rate to the number given if possible. All hardware interfaces do not support all speeds.

`ispeed 0 110 300 600 1200 1800 2400 4800 9600 19200 38400`
Sets terminal input baud rate to the number given. (Not all hardware supports split baud rates.) If the input baud rate is set to zero, the input baud rate is specified by the value of the output baud rate.

`ospeed 0 110 300 600 1200 1800 2400 4800 9600 19200 38400`
Sets terminal output baud rate to the number given, if possible. (Not all hardware supports split baud rates.) If the output baud rate is set to 0, the line will be hung up immediately.

`hupcl` (`-hupcl`) Hangs up (does not hang up) dataphone dataset connection on last close.

`hup` (`-hup`) Same as `hupcl` (`-hupcl`).

`cstopb` (`-cstopb`) Uses two (one) stop bits per character.

`cread` (`-cread`) Enables (disables) the receiver.

`clocal` (`-clocal`) Assumes a line without (with) modem control.

`loblk` (`-loblk`) Blocks (does not block) output from a noncurrent layer.

Input Modes

`ignbrk` (`-ignbrk`) Ignores (does not ignore) break on input.

`brkint` (`-brkint`) Signals (does not signal) interrupt on break.

`ignpar` (`-ignpar`) Ignores (does not ignore) parity errors.

`parmrk` (`-parmrk`) Marks (does not mark) parity errors. See `termio`(4).

`inpck` (`-inpck`) Enables (disables) input parity checking.

`istrip` (`-istrip`) Strips (does not strip) input characters to seven bits.

`inlcr` (`-inlcr`) Maps (does not map) `<newline>` to `<return>` on input.

`igncr` (`-igncr`) Ignores (does not ignore) CR on input.

`icrnl` (`-icrnl`) Maps (does not map) CR to NL on input.

`iuclc` (`-iuclc`) Maps (does not map) uppercase alphabetic characters to lowercase on input.

`ixon` (`-ixon`) Enables (disables) START/STOP output control. Output is stopped by sending an ASCII DC3 and started by sending an ASCII DC1.

`ixany` (`-ixany`) Allows any character (only DC1) to restart output.

`ixoff` (`-ixoff`) Requests that the system send (not send) START/STOP characters when the input queue is nearly empty or full.

`imaxbel` (`-imaxbel`)
: Echoes (does not echo) BEL when the input line is too long.

Output Modes

`opost` (`-opost`)
: Post-processes output (does not post-process output; ignores all other output modes).

`olcuc` (`-olcuc`)
: Maps (does not map) lowercase alphabetic characters to uppercase on output.

`onlcr` (`-onlcr`)
: Maps (does not map) NL to CR-NL on output.

`ocrnl` (`-ocrnl`)
: Maps (does not map) CR to NL on output.

`onocr` (`-onocr`)
: Does not (does) output CRs at column 0.

`onlret` (`-onlret`)
: On the terminal, NL performs (does not perform) the CR function.

`ofill` (`-ofill`)
: Uses fill characters (use timing) for delays.

`ofdel` (`-ofdel`)
: Uses delete characters (NULs) as fill characters.

`cr0 cr1 cr2 cr3`
: Selects style of delay for carriage returns. See `termio`(4).

`nl0 nl1`
: Selects style of delay for line-feeds. See `termio`(4).

`tab0 tab1 tab2 tab3`
: Selects style of delay for horizontal tabs. See `termio`(4).

`bs0 bs1`
: Selects style of delay for backspaces. See `termio`(4).

`ff0 ff1`
: Selects style of delay for form-feeds. See `termio`(4).

`vt0 vt1`
: Selects style of delay for vertical tabs. See `termio`(4).

Local Modes

`extproc` (`-extproc`)
: Enables (disables) external processing mode. When in external processing mode, much of the functionality of the tty driver is omitted (for example, line editing and echoing of typed data), because it is assumed that this functionality is being done externally. This is the mode that `telnetd`(8) uses when it is running with the Telnet Linemode option enabled. Disabling external processing notifies the `telnetd`(8) process, which then disables the line-mode option.

`isig` (`-isig`)
: Enables (disables) the checking of characters against the special control characters INTR, QUIT, and SWTCH.

`icanon` (`-icanon`)
: Enables (disables) canonical input (ERASE and KILL processing).

`xcase` (`-xcase`)
: Canonical (unprocessed) uppercase and lowercase presentation.

`echo` (`-echo`)
: Echoes back (does not echo back) every character typed.

`echoe` (`-echoe`)	Echoes (does not echo) ERASE character as a backspace-space-backspace string. Note: This mode erases the ERASEed character on many CRT terminals; however, it does not keep track of column position and, therefore, may be confusing on escaped characters, tabs, and backspaces.
`echok` (`-echok`)	Echoes (does not echo) NL after KILL character.
`lfkc` (`-lfkc`)	The same as `echok` (`-echok`); obsolete.
`echonl` (`-echonl`)	Echoes (does not echo) NL.
`noflsh` (`-noflsh`)	Disables (enables) flush after INTR, QUIT, or SWTCH.
`stwrap` (`-stwrap`)	Disables (enables) truncation of lines longer than 79 characters on a synchronous line.
`tostop` (`-tostop`)	Sends (does not send) SIGTTOU when background processes write to the terminal.
`echoctl` (`-echoctl`)	Echoes (does not echo) control characters as *char*, delete as ^?
`echoprt` (`-echoprt`)	Echoes (does not echo) erase character as character is *erased*.
`echoke` (`-echoke`)	BS-SP-BS erase (does not BS-SP-BS erase) entire line on line kill. Note: The `echoke` mode is not available in UNICOS.
`flusho` (`-flusho`)	Output is (is not) being flushed.
`pendin` (`-pendin`)	Retypes (does not retype) pending input at next read or input character.
`iexten` (`-iexten`)	Enables (disables) extended (implementation-defined) functions for input data.
`stflush` (`-stflush`)	Enables (disables) flush on a synchronous line after every `write`(2) system call.
`stappl` (`-stappl`)	Uses application mode (uses line mode) on a synchronous line.

Control Assignments

control-character c	Sets *control-character* to *c*; *control-character* is `ctab`, `discard`, `dsusp`, `eof`, `eol`, `eol2`, `erase`, `intr`, `kill`, `lnext`, `quit`, `reprint`, `start`, `stop`, `susp`, `swtch`, or `werase`. `ctab` is used with `-stappl`; see `termio`(4). If *c* is preceded by an (escaped from the shell) caret (^), the value used is the corresponding `CONTROL` character (for example, `^d` is `<CONTROL-d>`); `^?` is interpreted as `<DEL>`, and `^-` and `undef` are interpreted as {_POSIX_VDISABLE} if {_POSIX_VDISABLE} is in effect for the terminal.
`min`, `time` *number*	Sets the value of `min` or `time` to *number*. `min` and `time` are used in noncanonical mode input processing (`-icanon`).
`line` *i*	Sets line discipline to *i* ($0 < i < 127$).

Combination Modes

`evenp` or `parity`	Enables `parenb` and `cs7`.
`oddp`	Enables `parenb`, `cs7`, and `parodd`.
`-parity` or `-evenp`	Disables `parenb`, and set `cs8`.
`-oddp`	Disables `parenb` and `parodd`, and set `cs8`.
`-spacep`	Disables `parenb` and `parext`, and set `cs8`.
`-markp`	Disables `parenb`, `parodd`, and `parext`, and set `cs8`.
`raw` (`-raw` or `cooked`)	Enables (disables) raw input and output (no ERASE, KILL, INTR, QUIT, SWTCH, EOT, or output postprocessing).
`nl` (`-nl`)	Sets (unsets) `icrnl`. In addition, `-nl` unsets `inlcr` and `igncr`.
`lcase` (`-lcase`)	Sets (unsets) `xcase`, `iuclc`, and `olcuc`.
`LCASE` (`-LCASE`)	Same as `lcase` (`-lcase`).
`tabs` (`-tabs` or `tab3`)	Preserves (expands to spaces) tabs when printing.
`ek`	Resets ERASE and KILL characters back to `<CONTROL-u>` and `<DEL>`.
`sane`	Resets all modes to some reasonable values.

Window Size

`rows` *n*	Sets window size to *n* rows.
`columns` *n*	Sets window size to *n* columns.
`ypixels` *n*	Sets vertical window size to *n* pixels.
`xpixels` *n*	Sets horizontal window size to *n* pixels.

NOTES

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
`system`, `secadm`	In a privileged administrator shell environment, allowed to write shell-redirected output to any file.
`sysadm`	Shell-redirected output is subject to security label restrictions.

If the `PRIV_SU` configuration option is enabled, the super user can write shell-redirected output to any file.

EXIT STATUS

The `stty` utility exits with one of the following values:

0 The terminal options were read or set successfully.

>0 An error occurred.

EXAMPLES

Example 1: The following example reports certain terminal settings for the standard input device:

```
$ stty
speed 9600 baud; -parity hupcl -cread
rows = 40; columns = 80; ypixels = 0; xpixels = 0;
-inpck -istrip icrnl -ixany onlcr tab3
extproc echo echoe echok
```

Example 2: The following example reports all terminal settings for `/dev/tty`:

```
$ stty -a </dev/tty
speed 9600 baud; line = 0;
rows = 40; columns = 80; ypixels = 0; xpixels = 0;
intr = ^c; quit = ^; erase = ^?; kill = ^u;
eof = ^d; eol = ^@; eol2 = ^@; swtch = ^z;
start = ^q; stop = ^s; susp = ^z; dsusp = ^y;
rprnt = ^r; flush = ^o; werase = ^w; lnext = ^v;
-parenb -parodd cs8 -cstopb hupcl -cread -clocal -loblk
-ignbrk -brkint ignpar -parmrk -inpck -istrip -inlcr -igncr icrnl -iuclc
ixon -ixany -ixoff
extproc isig icanon -xcase echo echoe echok -echonl -noflsh
-tostop -iexten
opost -olcuc onlcr -ocrnl -onocr -onlret -ofill -ofdel tab3
```

Example 3: The following example turns off character echo on the current terminal:

```
$ stty -echo
$
```

Example 4: The following example reports the current settings for use on a later `stty` command line:

```
$ stty -g
d20:1805:4bd:13b:3:1c:8:15:4:0:0:1a
$
```

Example 5: The following example shows how to use the output of a previous `stty -g` command to set the current terminal:

```
$ stty d20:1805:4bd:13b:3:1c:8:15:4:0:0:1a
$
```

Example 6: The following example sets the current terminal characteristics to those of `/dev/ttyp003`:

```
$ stty ‘stty -g </dev/ttyp003‘
```

SEE ALSO

`ioctl`(2) to perform functions on character special files in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

`termio`(4) for information on general terminal interface in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

`telnetd`(8) to invoke the DARPA TELNET protocol server in the *UNICOS Administrator Commands Reference Manual*, Cray Research publication SR–2022

NAME

su – Lets you become another user or the super user

SYNOPSIS

su [-] [*name* [*args*]]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The su utility allows you to become another user without logging off. The default user *name* is root.

To use the su utility, the appropriate password must be supplied (unless you are an appropriately authorized user). If the password is correct, su will execute a new shell with the real and effective user ID set to that of the specified user. The new shell will be the optional program specified in the shell field of the specified user's password file entry (see udb(5)), or /bin/sh if none is specified (see sh(1)). To restore your original user identity, exit the new shell.

Any additional arguments specified on the command line are passed to the program invoked as the shell. For example, when sh(1) is used, an argument of the form -c *string* executes *string* via the shell and an option of -r will give the user a restricted shell.

The following statements are true only if the optional program specified in the shell field of the specified user's password file entry is like sh(1). If the first argument to su is -, the environment will be changed to what would be expected if the user actually logged in as the specified user. This is done by invoking the program used as the shell with an *arg0* value whose first character is -, causing the system's profile (/etc/profile) and then the specified user's profile (.profile in the new home directory) to be executed. Otherwise, the environment is passed along, with the possible exception of $PATH, which is set to /bin:/etc:/usr/bin for root. If the optional program used as the shell is /bin/sh, the user's .profile can check *arg0* for -sh or -su to determine whether it was invoked by login(1) or su, respectively. If the user's program is not /bin/sh, the program is invoked with an *arg0* of -*program* by both login(1) and su.

All attempts to become another user using su are logged in the log file /usr/adm/sulog. In addition, you are limited to two failed su attempts per minute. You are cautioned after the second failure, and logged off and disabled from relogging on after the third failure in any minute, and the setuid(2) system calls are logged in the security log.

The su utility invokes the centralized identification and authorization library routines to validate the user ID and password.

The su utility accepts the following options:

- Changes environment to that of specified user name.

name Indicates user name to which to log on (default is root).

args Specifies shell arguments for new login.

NOTES

If this utility is installed with a privilege assignment list (PAL), a user who is assigned the following privilege text upon execution of this command is allowed to perform the action shown:

Privilege Text	Action
`all`	Allowed to `su` to any user without supplying a password.

If this utility is installed with a PAL, a user with one of the following active categories is allowed to perform the action shown:

Active Category	Action
`system`, `secadm`	Allowed to `su` to any user without supplying a password.

If the `PRIV_SU` configuration option is enabled, the super user is allowed to `su` to any user without password.

CAUTIONS

If the `-` argument is used, the `TMPDIR` environment variable for the specified user is set to `JTMPDIR`, which is defined in `/usr/include/tmpdir.h`.

New limits are not set for the new user; the current limits are inherited.

EXAMPLES

Example 1: To become user `bin` while retaining your previously exported environment, enter the following:

```
su bin
```

Example 2: To become user `bin` but change the environment to what would be expected if `bin` had originally logged in, enter the following:

```
su - bin
```

Example 3: To execute *command* with the temporary environment and permissions of user `bin`, enter the following:

```
su - bin -c "command  arguments"
```

FILES

`/etc/udb`	User validation file containing user control limits
`/etc/profile`	System's start-up file for standard shell
`$HOME/.profile`	User's start-up file for standard shell
`/usr/adm/sulog`	Log file

SEE ALSO

`env`(1), `login`(1), `privtext`(1), `sh`(1), `tmpdir`(1)

`chown`(2), `setuid`(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

`ia_failure`(3C), `ia_mlsuser`(3C), `ia_success`(3C), `ia_user`(3C) in the *UNICOS System Libraries Reference Manual*, Cray Research publication SR–2080

`passwd`(5), `profile`(5), `udb`(5) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

General UNICOS System Administration, Cray Research publication SG–2301

NAME

sum – Prints checksum and block count of a file

SYNOPSIS

sum [-r] [*files*]

IMPLEMENTATION

All Cray Research systems

STANDARDS

XPG4

DESCRIPTION

The sum utility calculates and writes a 16-bit checksum for *files* and writes the number of 512-byte blocks in the files to the standard output. If no files are specified, data is read from standard input.

The sum utility accepts the following option and operand:

-r Causes an alternative algorithm to be used in computing the checksum.

files Specifies the files to be checked.

NOTES

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
system, secadm	Allowed to print checksum and size information for any file. In a privileged administrator shell environment, shell-redirected I/O is not subject to file protections.
sysadm	Allowed to print checksum and size information for any file subject to security label restrictions. Shell-redirected I/O is subject to security label restrictions.

If the PRIV_SU configuration option is enabled, the super user is allowed to print checksum and size information for any file. Shell-redirected I/O on behalf of the super user is not subject to file protections.

EXIT STATUS

The sum utility exits with one of the following values:

0 Successful completion.

>0 An error occurred.

MESSAGES

`Read error` Indistinguishable from end-of-file on most devices; check the block count.

SEE ALSO

`cksum`(1), `wc`(1)

NAME

sync – Flushes file system cache

SYNOPSIS

```
sync
```

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The sync utility executes the sync system primitive. It flushes all previously unwritten system buffers out of main memory. The buffers are written to disk unless a logical device cache is being used. In this instance, buffers are written to a solid-state storage device (SSD) and an ldsync(8) utility is required to flush the buffers to disk. If the system is to be stopped, sync(2) and ldsync(8) must be called to ensure file system integrity. See sync(2) for details.

SEE ALSO

sync(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

ldsync(8) in the *UNICOS Administrator Commands Reference Manual*, Cray Research publication SR–2022

NAME

sysconf – Displays system configuration data

SYNOPSIS

```
sysconf
```

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The sysconf utility displays hardware and software configuration data available from the sysconf(2) system call. The display uses the last part of the system call parameter as a keyword description, so that information as to the meaning of a field can be easily looked up in the sysconf(2) description. For example, the parameter _SC_CRAY_SYSTEM is displayed as SYSTEM= ???.

NOTES

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
system, secadm	In a privileged administrator shell environment, allowed to write shell-redirected output to any file.
sysadm	Shell-redirected output is subject to security label restrictions.

If the PRIV_SU configuration option is enabled, the super user can write shell-redirected output to any file.

SEE ALSO

getconf(1)

sysconf(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

NAME

tabs – Sets tabs on a terminal

SYNOPSIS

tabs [-T *term*] [-*n* | -a | -a2 | -c | -c2 | -c3 | -f | -p | -s | -u]

tabs [-T *term*] *n1*[,*n2*,...]

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The tabs utility sets the tab stops on the user's terminal according to the tab specification *tabspec*, after clearing any previous settings. The user's terminal must have hardware tabs that can be set remotely. The tabs utility accepts the following options:

Two types of tab specification are accepted for *tabspec*, either repetitive (-*n*) or arbitrary (*n1*,*n2*,...). If you omit *tabspec*, the default value is -8, (for example, UNIX system ''standard'' tabs). The lowest column number is 1. For tabs, column 1 always refers to the leftmost column on a terminal, even if column markers begin at 0.

-*n* A *repetitive* specification requests tabs at columns 1+ *n*, 1+2* *n*, and so on. The value 8 represents the UNIX system ''standard'' tab setting, and it is the most likely tab setting to be found at a terminal. The value 0 implies no tabs.

n1[, *n2*,...]
The *arbitrary* format permits the user to type any chosen set of numbers, separated by commas, in ascending order. Up to 40 numbers are allowed. If any number (except the first one) is preceded by a plus sign, it is considered an increment to be added to the previous value. Thus, the formats 1,10,20,30, and 1,10,+10,+10 are considered identical.

-T *term* The tabs utility usually must know the type of terminal to set tabs. *term* is a name listed in term(5). If you omit the -T option, tabs uses the value of the TERM environment variable. If TERM is not defined in the environment (see environ(7)), tabs tries a sequence that will work for GE Terminet 300 terminals.

The following options set tabs to some commonly used values:

-a 1,10,16,36,72
Assembler, format one.

-a2 1,10,16,40,72
Assembler, format two.

`-c` 1,8,12,16,20,55
COBOL, normal format.

`-c2` 1,6,10,14,49
COBOL, compact format one.

`-c3` 1,6,10,14,18,22,26,30,34,38,42,46,50,54,58,62,67
COBOL, compact format two.

`-f` 1,7,11,15,19,23
FORTRAN.

`-p` 1,5,9,13,17,21,25,29,33,37,41,45,49,53,57,61
PL/1.

`-s` 1,10,55
SNOBOL.

`-u` 1,12,20,44
Alternative assembler format.

To set tabs, use standard output.

NOTES

No consistency exists among different terminals to clear tabs.

This implementation relies entirely on information in the `terminfo` database for its operation.

EXIT STATUS

The `tabs` utility exits with one of the following values:

0 Successful completion.

>0 An error occurred.

SEE ALSO

`term`(5) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

`environ`(7) (available only online)

NAME

`tail` – Copies the last part of a file

SYNOPSIS

`tail` [`-f`] [`-c` *number*] [*file*]
`tail` [`-f`] [`-b` *number*] [*file*]
`tail` [`-f`] [`-n` *number*] [`-r`] [*file*]

Obsolescent version; may not be supported in future releases:

`tail` [±[*number*][*unit*][`r`][`f`]] [*file*]

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4
AT&T extensions (`-b` and `-r` options)

DESCRIPTION

The `tail` utility copies *file* to standard output, beginning at a designated place. If you omit *file*, standard input is used.

The `tail` utility accepts the following options:

`-f` With this option, if the input file is not a pipe, the program will not terminate after the line of the input file has been copied, but will enter an endless loop. It sleeps for a second and then attempts to read and copy more records from the input file. Thus, you can use it to monitor the growth of a file that is being written by some other process. If you omit *file*, and standard input is a pipe, this option is ignored.

`-c` *number* The *number* argument is a decimal integer whose sign affects the location in the file, measured in bytes, to begin the copying. If the sign is `+`, copying starts relative to the beginning of the file. If the sign is `-` or if you omit the sign, copying starts relative to the end of the file. The origin for counting is 1 (for example, `-c +1` represents the first byte of the file, `-c -1` the last byte of the file).

`-b` *number* This option is equivalent to `-c` *number*, except the starting location in the file is measured in 512-byte blocks rather than bytes.

`-n` *number* This option is equivalent to `-c` *number*, except the starting location in the file is measured in lines rather than bytes.

`-r` This option copies lines from the end of the file in reverse order. The default is to print the entire file in reverse order.

file File to be copied to standard output.

In the obsolescent version, an argument that begins with a `-` or `+` can be used as a single option. *unit* can be one of `b`, `c`, or `l`. The ±*number* argument with the letter `c` specified as a suffix is equivalent to `-c` ±*number*; ±*number* with the letter `l` specified as a suffix is equivalent to `-n` ±*number*; ±*number* with the letter `b` specified as a suffix is equivalent to `-b` ±*number*. If you omit *unit*, `n` is assumed. If you omit *number*, 10 is used. The letter `f` specified as a suffix is equivalent to specifying the `-f` option. Specifying the letter `r` alone or with the `l` suffix is equivalent to `-r`.

In the nonobsolescent form, if you omit `-b`, `-c`, or `-n`, `-n 10` is assumed.

NOTES

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
`system`, `secadm`	In a privileged administrator shell environment, shell-redirected I/O is not subject to file protections.
`sysadm`	Shell-redirected output is subject to security label restrictions.

If the `PRIV_SU` configuration option is enabled, shell-redirected I/O on behalf of the super user is not subject to file protections.

CAUTIONS

The `tail` utility copies only the last 32,768 bytes of a file, regardless of its line count.

EXIT STATUS

The `tail` utility exits with one of the following values:

0 Successful completion.

>0 An error occurred.

BUGS

Because `tail` commands that are relative to the end of the file are stored in a buffer, they are limited in length.

Various kinds of anomalous behavior may happen with character special files.

EXAMPLES

Example 1: The following command prints the last 10 lines of the file `fred`, followed by any lines that are appended to `fred` between the time `tail` is initiated and killed:

```
tail -f fred
```

Example 2: The following command prints the last 15 characters of the file `fred`, followed by any lines that are appended to `fred` between the time `tail` is initiated and killed:

```
tail -c 15 -f fred
```

SEE ALSO

`cat`(1), `head`(1), `more`(1), `pg`(1)

NAME

`talk` – Enables one user to communicate with another user

SYNOPSIS

`/usr/ucb/talk` *address* [*terminal*]

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The `talk` utility is a visual communication program that copies lines from your terminal to that of another user. This utility requires the TCP/IP networking software running under UNICOS.

The `talk` utility accepts the following operands:

address Specifies the login name of the person to whom you want to talk. If you want to talk to someone on your own system, use the person's login name for the *address* argument. If you want to talk to someone on a different host system, use one of the following formats for *address*:

user@host (preferred usage)
host!user
host:user

terminal Indicates the terminal name. If you want to talk to a user who is logged in more than once, use the *terminal* argument to indicate the appropriate terminal name.

When first called, the `talk` utility sends the following message to the person to whom you want to talk:

```
Message from TalkDaemon@his_machine...
talk: connection requested by your_name@your_machine.
talk: respond with: talk your_name@your_machine
```

At this point, the message recipient should reply by entering the following:

```
talk your_name@your_machine
```

It does not matter from which machine the recipient replies, as long as the login name is the same. When communication is established, the two parties may type simultaneously, with their output appearing in separate windows. If you press <CONTROL-l>, the screen is reprinted; the erase, kill, and word-kill characters work normally. If you press <CONTROL-g>, an <alert> character is sent to both terminals. To exit the talk utility, type an interrupt character; the cursor moves to the bottom of the screen and the command restores the terminal.

Users may deny or grant other users the permission to talk to them by using the mesg(1) command. At the outset, the use of talk is allowed. Certain utilities, in particular pr(1), disallow messages to prevent messy output.

NOTES

The talk utility requires the TCP/IP networking software to run under UNICOS.

EXIT STATUS

The talk utility exits with one of the following values:

0 Successful completion.

>0 An error occurred, or talk was invoked on a terminal incapable of supporting it.

FILES

/etc/hosts	Finds the recipient's machine
/etc/utmp	Finds the recipient's tty

SEE ALSO

mail(1), mesg(1), pr(1), who(1), write(1)

hosts(5), utmp(5) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

NAME

`tar` – Archives tape files

SYNOPSIS

`tar -c`[*modifiers*] [*files*]
`tar -r`[*modifiers*] [*files*]
`tar -t`[*modifiers*] [*files*]
`tar -u`[*modifiers*] [*files*]
`tar -x`[*modifiers*] [*files*]

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The `tar` utility saves and restores files on magnetic tape and disk files.

Note: The `tar` options can be followed by zero or more *modifiers*. For descriptions of the *modifiers*, see the subsection following the option descriptions.

The `tar` utility supports the following options:

`-c` (Create) Creates a new archive; writing begins at the beginning of the archive, instead of after the last file.

`-r` (Replace) Writes the specified *files* on the end of the archive. This option does not work with tape devices.

`-t` (Table) Lists the names and other information for the specified files each time that they occur on the archive. The listing is similar to the format produced by the `ls -l` command. If you omit *files*, all the names on the archive are listed.

`-u` (Update) Adds the specified *files* to the archive if they are not already there or have been modified since last written on that archive. This option implies the `-r` option.

`-x` (Extract) Extracts the specified *files* from the archive. If a specified file matches a directory whose contents had been written onto the archive, this directory is (recursively) extracted. You must use the file or directory's relative path when appropriate; otherwise `tar` will not find a match. The owner, modification time, and mode are restored (if possible). If you omit *files*, the entire content of the archive is extracted. If several files with the same name are on the archive, the last one overwrites all earlier ones.

The tar options can be followed immediately by zero or more of the following *modifiers*:

a Excludes copy or preservation of access control lists (ACLs). The a modifier is useful only with the s modifier.

b (Blocking factor) Causes tar to use the *block* argument as the blocking factor for (tape) records in the archive. The default is 20 for tape archives and 128 for disk archives. The maximum is 128. The size of a block is 512 bytes. The block size is determined automatically when reading tapes created on block special devices (-x and -t options). The blocking factor should match the argument given to the -b option of the tpmnt(1) utility for archives written to tape.

f (File) Causes tar to use the next argument as the name of the archive. If you omit the f modifier, tar uses the default, which is /dev/extape. If the name of the file is -, tar writes to the standard output or reads from the standard input, whichever is appropriate. Thus, you can use tar as the head or tail of a pipeline. You can also use tar to move hierarchies with the utility, as follows:

```
cd fromdir; tar -cf - . | (cd todir; tar -xf -)
```

h Follows symbolic links as if they were normal files or directories. Usually, tar does not follow symbolic links.

l (Link) tar sends an error message if it cannot resolve all the links to the files being dumped. If you omit the l modifier, no error messages are printed.

m (Modify) tar does not restore the modification times. If you use the m modifier, the modification time of the file will be the time of extraction. The m modifier is valid only with the -x option.

o (Ownership) Causes extracted files to take on the user and group identifier of the user running the program, rather than those on tape. This option is always on, unless the O modifier is used, and is valid only with the -x option.

O (Ownership) Turns off the o modifier. This causes extracted files to take on the user and group identifier from the tape, rather than those of the user running the program. Users must have chown(2) permission in their UDB record to use this option.

p (Permissions) Preserves the original file permissions on extracted files. Clears the umask of the process that extracts the files. The p modifier applies only to the -x option.

s (Secure) Performs a secure copy (security information and ACLs).

v (Verbose) Usually, tar does its work silently. The v modifier causes tar to display the name of each file it treats, preceded by the option. With the -t option, the v modifier gives more information about the tape entries than the ls(1) command.

w (What) Causes tar to display the action to be taken, followed by the name of the file, and then to wait for your confirmation. If you begin a word with y, the action is performed. Any other input means ‘‘no.’’ The w modifier is not valid with the -t option.

The `tar` utility supports the following operand:

files Specifies which files (or directories) are to be dumped or restored. In all cases, appearance of a directory name refers to the files and (recursively) subdirectories of that directory.

A secure save (`s` modifier) outputs security information and access control lists (ACLs) corresponding to regular files. A secure restore (`s` modifier) installs security information and ACLs of regular files.

NOTES

By default, when extracting files, the modes of the files will be set using the present `umask`. To preserve the original modes, you can invoke `tar` using the `p` modifier.

You may copy only files that have the same security label as the user.

A user's active categories must dominate the categories of the file being copied.

The `tar` utility saves information about regular files. Thus, security information and ACLs for directories are not preserved.

ACLs are preserved only if the user is the owner of the file, or has an active `secadm` category, or has both read and write access to the file being copied.

CAUTIONS

It is recommended that you use a blocking factor of 8 on all tapes. For example, the following creates a tape with a blocking factor of 8:

```
tar -cb 8
```

MESSAGES

Reports non-valid options and tape read and write errors.
Reports not enough memory available to hold the link tables.

BUGS

You cannot request the *n*th occurrence of a file.

The `-u` option can be slow.

You must not use the `b` modifier with archives that will be updated. The current magnetic tape driver cannot backspace raw magnetic tape. If the archive is on a disk file, you must not use the `b` modifier, because updating an archive stored on disk can destroy it.

The length of a file name is currently limited to 100 characters.

The `tar` utility does not copy empty directories or special files.

EXAMPLES

Example 1: The following command backs up a user's entire directory to online magnetic tape:

```
cd
rsv CART 1
tpmnt -l nl -v vsn -P tapefile -b 4096 -g CART -n
tar -cvfb tapefile 8 .
rls -a
```

The *vsn* variable is the volume serial number of the tape. *tapefile* is the path name for the tape being used. A blocking factor of 8 (4096 bytes) was chosen so that the tape can be read back with online tape.

Example 2: The following example demonstrates how you can use `tar` to write and then read an online tape file of blocking factors other than 8:

```
rsv
tpmnt -l nl -p tapefile -b 512 -v vsn
tar -cvfb tapefile 1 .
cd newdir
tar -xvf tapefile
rls -a
```

Example 3: This example shows how to read a `tar` tape from another UNIX system that was written with a blocking factor of 20:

```
rs
tpmnt -l nl -p tapefile -b 10240 -v vsn
tar -xvf tapefile
rls -a
```

FILES

`/dev/extape`	External tape file
`/tmp/tar*`	Work files

SEE ALSO

`ar`(1), `cpio`(1), `ls`(1), `rls`(1), `rsv`(1), `tpmnt`(1)

`chown`(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

General UNICOS System Administration, Cray Research publication SG–2301

NAME

`target` – Verifies target CPU characteristics

SYNOPSIS

`target` [`-s`] [*cpuname*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `target` utility determines and prints the CPU characteristic for the machine type specified by *cpuname*.

The `target` utility accepts the following options:

`-s` Prints only the machine subtype field. The machine subtype field may have one of the following values:

Machine Subtype	Description
`CRAY-C90`	CRAY C90 series
`CRAY-EL`	CRAY EL series
`CRAY-J90`	CRAY J90 series
`CRAY-JSE`	CRAY J90se series
`CRAY-TS`	CRAY T90 series
`CRAY-TS-IEEE`	CRAY T90 series with IEEE floating point
`CRAY-T3D`	CRAY T3D systems
`CRAY-T3E`	CRAY T3E systems
`CRAY-YMP`	CRAY Y-MP model 832

If the `-s` option is not specified, the `target` utility prints the subtype field without the `CRAY-` prefix. This is printed when the target is a Cray PVP system.

cpuname Specifies a Cray Research machine type. The *cpuname* argument can be any one of the following (uppercase or lowercase):

Machine Type	Description
`cray-c90`	CRAY C90 series
`cray-el`	CRAY EL series
`cray-j90`	CRAY J90 series
`cray-jse`	CRAY J90se series
`cray-ts`	CRAY T90 series
`cray-t3d`	CRAY T3D systems
`cray-t3e`	CRAY T3E systems
`cray-ymp`	CRAY Y-MP model 832
`host`	
`\*host`	

```
iop
target          Default
\*target
```

If *cpuname* is `target`, the current environment is checked for the `TARGET` environment variable. If it is found, `target` parses the environment variable and displays the machine characteristics for the target machine. If *cpuname* is `host`, the host machine characteristics are displayed. For any other specified machine type, the machine type's default characteristics are given.

The `target` utility does not make changes to the current `TARGET` environment variable. The user must initialize and/or make changes to this environment variable at the shell level when necessary. The `target` utility verifies that the environment variable is syntactically correct.

To initialize the `TARGET` environment variable, enter the following in the standard shell:

```
export TARGET
```

The format to set up or change the `TARGET` environment variable in the standard shell is as follows:

```
TARGET=[cpuname]{,[charac]}
```

The target machine represented by `TARGET` takes on the default machine characteristics specified by *cpuname*, modified accordingly by any specified *charac* arguments. If you do not specify *cpuname*, the machine characteristics of the host machine are used and modified. On CRAY T3E systems, the characteristics of the PE on which the `target` utility is executing are returned.

The *cpuname* and *charac* arguments on the `TARGET` variable can be the following:

cpuname Same options as listed previously.

charac These are possible features that may be specified for the given *cpuname* computer. In some cases, it is possible to choose characteristics that may not make sense for a given *cpuname* computer. You cannot specify characteristics for `iop`.

All Cray Research systems let you specify the following numerical trait:

Numeric Trait	**Description**
`memsize=`*n*[*y*]	Memory size in words. Using `k` for *y* defines (*n* * 1024) words, using `m` for *y* defines (*n* * 1,048,576) words.

Cray PVP systems also let you specify the following numerical traits:

Numeric Trait	**Description**
`banks=`*n*	Number of memory banks.
`numcpus=`*n*	Number of CPUs.
`ibufsize=`*n*	Instruction buffer size.
`memspeed=`*n*	Memory speed in clock periods.
`clocktim=`*n*	Clock period in picoseconds.

`numclstr=`*n* Number of clusters.

`bankbusy=`*n* Number of clock periods that the memory bank reserved.

Cray PVP systems (except CRAY T90 series where noted) let you specify the following logical traits:

Logical Trait	Description
`bmm`	Bit matrix multiply unit.
`nobmm`	No bit matrix multiply unit (not valid for `cray-ts`).
`ema`	Extended memory addressing for 24-bit mode (not valid for `cray-ts`).
`noema`	No extended memory addressing for 24-bit mode.
`cigs`	Compressed index and gather/scatter.
`nocigs`	No compressed index or gather/scatter.
`vpop`	Vector pop count.
`novpop`	No vector pop count (not valid for `cray-ts`).
`pc`	Programmable clock.
`nopc`	No programmable clock (not valid for `cray-ts`).
`readvl`	Read vector length.
`noreadvl`	Do not read vector length (not valid for `cray-ts`).
`vrecur`	Vector recursion (not valid for `cray-ts`).
`novrecur`	No vector recursion.
`avl`	Additional vector logical.
`noavl`	No additional vector logical (not valid for `cray-ts`).
`hpm`	Hardware performance monitor.
`nohpm`	No hardware performance monitor (not valid for `cray-ts`).
`statrg`	Status register.
`nostatrg`	No status register (not valid for `cray-ts`).
`bdm`	Bidirectional memory.
`nobdm`	No bidirectional memory (not valid for `cray-ts`).
`cori`	Control operand range interrupts.
`nocori`	No control operand range interrupts (not valid for `cray-ts`).
`addr32`	32-bit mode addressing (not valid for `cray-ts`).
`noaddr32`	No 32-bit mode addressing.

`xea`	CRAY Y-MP instruction timings (not valid for `cray-ts`).
`noxea`	No CRAY Y-MP instruction timings.
`avpop`	Additional vector pop count.
`noavpop`	No additional vector pop count.
`ieee`	IEEE floating-point arithmetic (not valid for `cray-ymp` or `cray-c90`).
`noieee`	No IEEE floating-point arithmetic.

NOTES

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
`system`, `secadm`	In a privileged administrator shell environment, allowed to write shell-redirected output to any file.
`sysadm`	Shell-redirected output is subject to security label restrictions.

If the `PRIV_SU` configuration option is enabled, the super user can write shell-redirected output to any file.

SEE ALSO

`exec`(2), `target`(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

`profile`(5) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

`GETPMC`(3F) in the *Application Programmer's Library Reference Manual*, Cray Research publication SR–2165

NAME

tbl – Formats tables for nroff(1) or troff(1)

SYNOPSIS

tbl [-TX] [*filename*] ...

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The tbl preprocessor prepares tables for nroff(1) or troff(1). tbl assumes that lines between the .TS and .TE command lines describe tables; thus they are reformatted. Lines outside these command lines are copied to the standard output. tbl does not alter the .TS and .TE command lines.

If no arguments are given, tbl reads the standard input, so tbl may be used as a filter. When tbl is used with eqn(1) or neqn(1), the tbl command should be first, to minimize the volume of data passed through pipes.

The tbl preprocessor accepts the following options:

-TX Forces tbl to use full vertical line motions. This option makes the output more suitable for devices that cannot generate partial vertical line motions (for example, line printers).

filename Specifies the files to be formatted.

The tbl preprocessor also accepts the following global options:

center Centers the table (default is left-adjust); expand makes the table as wide as the current line length.

box Encloses the table in a box.

doublebox Encloses the table in a double box.

allbox Encloses each item of the table in a box.

tab (*x*) Uses the character *x* instead of a tab to separate items in a line of input data.

linesize (*n*) Sets line or rules (for example, from box) in *n*-point type.

End the global options, if any, with a semicolon (;).

After global options come lines describing the format of each line of the table. Each such format line describes one line of the table itself, except that the last format line (which you must end with a period) describes *all* remaining lines of the table. A single key letter describes each column of each line of the table. You can follow this key letter with specifiers that determine the font and point size of the corresponding item, that indicate where vertical bars are to appear between columns, and that determine column width, intercolumn spacing, and so on. The available key letters are as follows:

c Centers item within the column.

r Right-justifies item within the column.

l Left-justifies an item within the column.

n Numerically adjusts item in the column: units positions of numbers are aligned vertically.

s Spans previous item on the left into this column.

a Centers longest line in this column and then left-justifies all other lines in this column with respect to that centered line.

^ Spans down previous entry in this column.

_ Replaces this entry with a horizontal line.

= Replaces this entry with a double horizontal line.

The characters B and I stand for the bold and italic fonts, respectively; the character | indicates a vertical line between columns.

The format lines are followed by lines containing the actual data for the table, followed finally by .TE. Within such data lines, data items are usually separated by tab characters.

If a data line consists of only _ or =, a single or double line, respectively, is drawn across the table at that point; if a *single item* in a data line consists of only _ or =, then that item is replaced by a single or double line. Some printers do not have the vertical resolution to produce double lines.

EXAMPLES

The following tbl example shows a simple three-column table. The characters \t represent a tab; when entering the text, type a genuine tab character:

```
.TS
c s s
c c s
c c c
l n n.
Household\tPopulation
Town\tHouseholds
\tNumber\tSize
Bedminster\t789\t3.26
Bernards Twp.\t3087\t3.74
Bernardsville\t2018\t3.30
Bound Brook\t3425\t3.04
Branchburg\t1644\t3.49
Bridgewater\t7897\t3.81
Far Hills\t240\t3.19
.TE
```

This input produces the following formatted table:

Town	Household Population	
	Households	
	Number	Size
Bedminster	789	3.26
Bernards Twp.	3087	3.74
Bernardsville	2018	3.30
Bound Brook	3425	3.04
Branchburg	1644	3.49
Bridgewater	7897	3.81
Far Hills	240	3.19

SEE ALSO

eqn(1), nroff(1), troff(1)

NAME

tc – troff(1) output interpreter

SYNOPSIS

tc [-t] [-o *list*] [-a *n*] [-e] [*file*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The tc utility interprets its input (standard input default) as output from troff(1). The standard output of tc is intended for a TEKTRONIX 4015 (a 4014 terminal with ASCII and APL character sets). The various typesetter sizes are mapped into the 4014's four sizes. The entire troff(1) character set is drawn using the 4014's character generator, using overstruck combinations where necessary.

The tc utility accepts the following options:

-t Does not wait between pages (for directing output into a file).

-o *list* Prints only the pages enumerated in *list*. The list consists of pages and page ranges (such as 5–17) separated by commas. The range *n*- goes from *n* to the end. The range -*n* goes from the beginning to and including page *n*.

-a *n* Sets the aspect ratio to *n*. The default is 1.5.

-e Does not erase before each page.

file Specifies the file to be interpreted.

A typical usage of tc follows:

```
troff file | tc
```

At the end of each page, tc waits for a new line (empty line) from the keyboard before continuing on to the next page. In this wait state, the following commands are recognized:

!*cmd* Sends *cmd* to the shell.

e Inverts the state of the screen erase.

-*n* Skips backward *n* pages.

a*n* Sets the aspect ratio to *n*.

? Prints a list of available options.

BUGS

When using `tc`, font distinctions are lost.

The `tc` command needs a `-w` option to wait for input to arrive.

SEE ALSO

`nroff`(1), `troff`(1)

NAME

`tee` – Duplicates output

SYNOPSIS

`tee` [`-a`] [`-i`] [*files*]

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The `tee` utility is a filter that copies standard input to standard output, making a copy in zero or more files. `tee` does not buffer its output.

The `tee` utility accepts the following options and operands:

`-a` Appends the output to each respective *file* rather than overwriting it.

`-i` Ignores the `SIGINT` signal.

files Files to be duplicated.

NOTES

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
`system`, `secadm`	In a privileged administrator shell environment, shell-redirected I/O is not subject to file protections.
`sysadm`	Shell-redirected output is subject to security label restrictions.

If the `PRIV_SU` configuration option is enabled, shell-redirected I/O on behalf of the super user is not subject to file protections.

EXIT STATUS

The `tee` utility exits with one of the following values:

0 No standard input was successfully copied to all output files.

>0 An error occurred.

EXAMPLES

The `tee` utility often is used in shell scripts to redirect data to more than one place simultaneously. In this example, you can simultaneously view the output of the `news`(1) utility and save the information in a file.

```
news | tee mynews
```

SEE ALSO

`sh`(1)

NAME

telnet – User interface to the TELNET protocol

SYNOPSIS

/usr/ucb/telnet [-8] [-E] [-K] [-L] [-S *tos*] [-X *atype*] [-a] [-c] [-d] [-e *escapechar*] [-k *realm*] [-l *user*] [-n *tracefile*] [-r] [-x] *host* [*port*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The telnet utility communicates with another host by using TELNET protocol. If you invoke telnet without the *host* argument, it enters command mode, indicated by its prompt (telnet>). In this mode, it accepts and executes the commands listed following. If you invoke telnet with arguments, it performs an open command with those arguments.

The telnet utility accepts the following options:

-8 Specifies an 8-bit data path. This causes an attempt to negotiate the TELNET BINARY option on both input and output.

-E Stops any character from being recognized as an escape character.

-K Specifies no automatic login to the remote system.

-L Specifies an 8-bit data path on output. This causes the BINARY option to be negotiated on output.

-S *tos* Sets the IP Type-of-Service (TOS) option for the telnet connection to the value *tos*, which can be a numeric TOS value or a symbolic TOS name found in the /etc/iptos file.

-X *atype* Disables the atype type of authentication.

-a Specifies automatic login. See the toggle autologin command on this man page for more information.

-c Disables the reading of the user's .telnetrc file. (See the toggle skiprc command on this man page.)

-d Sets the initial value of the debug toggle to TRUE.

-e *escapechar*
Sets the initial telnet esape character to *escapechar*. If you omit *escapechar*, there is no escape character.

-k *realm* If Kerberos authentication is being used, the -k option requests that telnet obtain tickets for the remote host in realm *realm* instead of the remote host's realm, as determined by krb_realmofhost(3K).

`-l` *user* When connecting to the remote system, if the remote system understands the TELNET `ENVIRON` option, *user* is sent to the remote system as the value for the TELNET environment variable `USER`. This option implies the `-a` option. You also can use this option with the `open` command.

`-n` *tracefile* Opens *tracefile* for recording trace information. (See the `set tracefile` command).

`-r` Specifies a user interface similar to `rlogin`(1B). In this mode, the escape character is set to the tilde (~) character, unless modified by the `-e` option.

`-x` Turns on encryption of the data stream if possible. This option is not available outside the United States and Canada.

host Indicates the official name, an alias, or the Internet address of a remote host.

port Indicates a port number (address of an application). If you do not specify a number, the default `telnet` port is used. Port names are mapped to port numbers through the `/etc/services` file. Usually, when you specify a port number, `telnet` does not send out any initial TELNET option negotiation. If the port number/name is preceded by a minus sign, the initial TELNET option negotiation is sent.

When in `rlogin` mode, a line of the form `~.` disconnects from the remote host; `~` is the `telnet` escape character. Similarly, the line `~^Z` suspends the `telnet` session. The line `~^]` escapes to the normal `telnet` escape prompt.

After a connection has been opened, `telnet` attempts to enable the TELNET `LINEMODE` option. If this fails, `telnet` reverts to one of two input modes: either character-at-a-time or line-by-line, depending on what the remote system supports.

When `LINEMODE` is enabled, character processing is done on the local system, under the control of the remote system. When input editing or character echoing will be disabled, the remote system relays that information. The remote system also relays changes to any special characters that occur on the remote system, so that they can become available on the local system.

Using the character-at-a-time mode, most text typed is immediately sent to the remote host for processing.

Using line-by-line mode, all text is echoed locally, and (usually) only completed lines are sent to the remote host. You can use the local echo character (initially ^E) to turn off and on the local echo (used to enter a password without the password being echoed).

If the `LINEMODE` option is enabled, or if the `localchars` toggle is `TRUE` (the default for line-by-line), the user's `quit`, `intr`, and `flush` characters are trapped locally, and they are sent as TELNET protocol sequences to the remote side. If `LINEMODE` was ever enabled, the user's `susp` and `eof` are also sent as TELNET protocol sequences, and `quit` is sent as a TELNET `ABORT` rather than `BREAK`. There are options (see `toggle autoflush` and `toggle autosynch` on this man page) that cause this action to flush subsequent output to the terminal (until the remote host acknowledges the TELNET sequence) and to flush previous terminal input (see `quit` and `intr`).

While connected to a remote host, you can enter `telnet` command mode by typing the `telnet` escape character (initially ^]). In command mode, the usual terminal editing conventions are available.

The following `telnet` commands are available: type only enough of each command to identify it uniquely (also do this for arguments to the `mode`, `set`, `toggle`, `unset`, `slc`, and `display` commands).

`auth` *arguments...*
The `auth` command manipulates the information sent through the TELNET `AUTHENTICATE` option. Valid arguments for the `auth` command are as follows:

`disable` *type* Disables the specified type of authentication. To obtain a list of available types, use the `auth disable?` command.

`enable` *type* Enables the specified type of authentication. To obtain a list of available types, use the `auth enable?` command.

`status` Lists the current status of the various types of authentication.

`close` Closes a TELNET session and returns to command mode.

`display` [*argument...*]
Displays all, or some, of the `set` and `toggle` values.

`encrypt` *arguments...*
The `encrypt` command manipulates the information sent through the TELNET `ENCRYPT` option.

Note: Because of export controls, Cray Research does not support the TELNET `ENCRYPT` option outside the United States and Canada.

Valid arguments for the `encrypt` command are as follows:

`disable` *type*[`input|output`]
Disables the specified type of encryption. If you omit `input` and `output`, both input and output are disabled. To obtain a list of available types, use the `encrypt disable?` command.

`enable` *type*[`input|output`]
Enables the specified type of encryption. If you omit `input` and `output`, both input and output are enabled. To obtain a list of available types, use the `encrypt enable?` command.

`input` This is the same as the `encrypt start input` command.

`-input` This is the same as the `encrypt stop input` command.

`output` This is the same as the `encrypt start output` command.

`-output` This is the same as the `encrypt stop output` command.

`start`[`input`|`output`]
: Attempts to start encryption. If you omit `input` and `output`, both input and output are enabled. To obtain a list of available types, use the `encrypt enable?` command.

`status` Lists the current status of encryption.

`stop`[`input`|`output`]
: Stops encryption. If you omit `input` and `output`, encryption is on both input and output.

`type` *type* Sets the default *type* of encryption to be used with later `encrypt start` or `encrypt stop` commands.

`environ` [*argument...*]
: The `environ` command manipulates the variables sent through the TELNET `ENVIRON` option. The initial set of variables is taken from the user's environment, with only the `DISPLAY` and `PRINTER` environment variables being exported by default. If you use the `-a` or `-l` options, the `USER` environment variable is also exported. Valid arguments for the `environ` command are as follows:

`define` *variable value*
: Defines the variable *variable* to have the value *value*. Any variables that this command defines are exported automatically. Enclose *value* in single or double quotation marks to include tabs and spaces.

`export` *variable*
: Marks the variable *variable* to be exported to the remote side.

`list` Lists the current environment variables. Those marked with a `*` are sent automatically; other variables are sent only if explicitly requested.

`undefine` *variable*
: Removes *variable* from the list of environment variables.

`unexport` *variable*
: Marks the variable *variable* to not be exported unless explicitly asked for by the remote side.

`?` Prints out help information for the `environ` command.

`logout` Sends the TELNET `LOGOUT` option to the remote side. This command is similar to a `close` command; however, if the remote side does not support the `LOGOUT` option, nothing happens. If, however, the remote side does not support the `LOGOUT` option, this command should cause the remote side to close the TELNET connection. If the remote side also supports the concept of suspending a user's session for later reattachment, the `logout` argument indicates that you should terminate the session immediately.

`mode` *type* The *type* option is one of several options to select; consider the state of the TELNET session when selecting. The remote host is asked for permission to go into the requested mode. If the remote host can enter that mode, the requested mode is entered.

`character` Disables the TELNET `LINEMODE` option, or, if the remote side does not understand the `LINEMODE` option, it enters the character-at-a-time mode.

`line` Enables the TELNET `LINEMODE` option, or, if the remote side does not understand the `LINEMODE` option, it attempts to enter line-by-line mode.

`isig` (`-isig`)
Attempts to enable (disable) the `TRAPSIG`1 mode of the `LINEMODE` option. This requires that you enable the `LINEMODE` option.

`edit` (`-edit`)
Attempts to enable (disable) the `EDIT` mode of the `LINEMODE` option. This requires that you enable the `LINEMODE` option.

`softtabs(-softtabs)`
Attempts to enable (disable) the `SOFT_TAB` mode of the `LINEMODE` option. This requires that you enable the `LINEMODE` option.

`litecho(-litecho)`
Attempts to enable (disable) the `LIT_ECHO` mode of the `LINEMODE` option. This requires that you enable the `LINEMODE` option.

`?` Prints out help information for the `mode` command.

`open` *host* [[`-q`]*port*]
Opens a connection to the specified host. If you do not specify a port number, `telnet` attempts to contact a TELNET server at the default port. The host specification can be either a host name (see `hosts`(5)) or an Internet address that is specified in the dot notation (see `inet`(3C)). You can use the `-l` option to specify the user name to be passed to the remote system through the `ENVIRON` option. The `-q` option can perform automatic login (see the `toggle autologin` command on this man page). When connecting to a nonstandard port, `telnet` omits any automatic initiation of TELNET options. When the port number is preceded by a minus sign, the initial option is negotiated. After establishing a connection, if the `skiprc` variable is not enabled (see the `toggle skiprc` command on this man page), the `.telnetrc` file in the user's home directory is opened. Lines that begin with a # symbol are comment lines. Blank lines are ignored. Lines that begin without white space are the start of a machine entry. The first item on the line is the name of the machine to which a connection is being made. The remainder of the line, and successive lines that begin with white space are assumed to be `telnet` commands and are processed as if they are typed manually in the `telnet` command prompt.

The special machine name `DEFAULT` specifies commands that should be executed for all machines. There can be more than one entry for a machine; all matches with the specified machine name will be executed.

`quit` Closes any open TELNET session and exits `telnet`. An end-of-file command (in command mode) also closes a session and exits.

`send` *arguments*

Sends one or more special character sequences to the remote host. You can specify the following arguments (you can specify more than one argument at a time):

`abort`	Sends the TELNET ABORT (ABORT processes) sequence.
`ao`	Sends the TELNET AO (Abort Output) sequence, which should cause the remote system to flush all output from the remote system to the user's terminal.
`ayt`	Sends the TELNET AYT (Are You There) sequence, to which the remote system might or might not choose to respond.
`brk`	Sends the TELNET BRK (Break) sequence, which can have significance to the remote system.
`ec`	Sends the TELNET EC (Erase Character) sequence, which should cause the remote system to erase the last character entered.
`el`	Sends the TELNET EL (Erase Line) sequence, which should cause the remote system to erase the line being entered currently.
`eof`	Sends the TELNET EOF (end-of-file) sequence.
`eor`	Sends the TELNET EOR (end-of-record) sequence.
`escape`	Sends the current `telnet` escape character (initially ^]).
`ga`	Sends the TELNET GA (Go Ahead) sequence, which probably has no significance to the remote system.
`getstatus`	Sends the TELNET STATUS SEND suboption if the remote side understands the TELNET STATUS option. The response is not seen unless the `options` variable was set.
`ip`	Sends the TELNET IP (Interrupt Process) sequence, which should cause the remote system to abort the currently running process.
`nop`	Sends the TELNET NOP (No OPeration) sequence.
`susp`	Sends the TELNET SUSP (SUSPend process) sequence.
`synch`	Sends the TELNET SYNCH sequence, which causes the remote system to discard all previously typed (but not yet read) input. This sequence is sent as TCP urgent data (and may not work if the remote system is a 4.2BSD system; if it does not work, a lowercase `r` might be echoed on the terminal).
`?`	Prints out help information for the `send` command.

`set` *argument value*

`unset` *arguments...*

The `set` command sets any one of several `telnet` variables to a specific value or to `TRUE`. The special value `off` turns off the function associated with the variable; this is equivalent to using the `unset` command, which disables or sets to `FALSE` any of the specified functions. Use the `display` command to interrogate variables. The variables that can be set or unset, but not toggled, are listed following. Also, any of the variables for the `toggle` command can be explicitly set or unset by using the `set` and `unset` commands.

`ayt` If TELNET is in `localchars` mode, or `LINEMODE` is enabled, and the `status` character is typed, a TELNET AYT sequence (see `send ayt` preceding) is sent to the remote host. The initial value for the `Are You There` character is the terminal's `status` character.

`echo` If in line-by-line mode, this is the value (initially `^E`) that toggles between doing local echoing of entered characters (for normal processing), and suppressing echoing of entered characters (for example, for entering a password).

`eof` If TELNET is operating in `LINEMODE` or line-by-line mode, entering this character as the first character on a line sends this character to the remote system. The initial value of the EOF character is the terminal's `eof` character.

`erase` If TELNET is in *localchars* mode (see `toggle localchars` on this man page), and if `telnet` is operating in character-at-a-time mode, when this character is typed, a TELNET EC sequence (see `send ec` listed previously) is sent to the remote system. The initial value for the erase character is the terminal's `erase` character.

`escape` TELNET escape character (initially ^[), which causes entry into `telnet` command mode (when connected to a remote system).

`flushoutput`
If TELNET is in *localchars* mode (see `toggle localchars` on this man page) and the `flushoutput` character is typed, a TELNET AO sequence (see `send ao` listed previously) is sent to the remote host. The initial value for the flush character is the terminal's `flush` character.

`forw1`
`forw2` If TELNET is operating in `LINEMODE`, these are the characters that, when typed, cause partial lines to be forwarded to the remote system. The initial value for the forwarding characters are taken from the terminal's `eol` and `eol2` characters.

`interrupt` If TELNET is in *localchars* mode (see `toggle localchars` on this man page) and the `interrupt` character is typed, a TELNET IP sequence (see `send ip` listed previously) is sent to the remote host. The initial value for the interrupt character is the terminal's `intr` character.

`kill` If TELNET is in *localchars* mode (see `toggle localchars` on this man page), and if `telnet` is operating in character-at-a-time mode, when this character is typed, a TELNET EL sequence (see `send el` listed previously) is sent to the remote system. The initial value for the kill character is the terminal's `kill` character.

`lnext` If TELNET is operating in `LINEMODE` or line-by-line mode, this character is the terminal's `lnext` character. The initial value for the `lnext` character is the terminal's `lnext` character.

`quit` If TELNET is in *localchars* mode (see `toggle localchars` on this man page) and the `quit` character is typed, a TELNET BRK sequence (see `send brk` listed previously) is sent to the remote host. The initial value for the quit character is the terminal's `quit` character.

`reprint` If TELNET is operating in `LINEMODE` or line-by-line mode, this character is the terminal's `reprint` character. The initial value for the reprint character is the terminal's `reprint` character.

`rlogin` This is the `rlogin` escape character. If set, the normal TELNET escape character is ignored unless it is preceded by this character at the beginning of a line. This character, at the beginning of a line followed by a `.` character closes the connection; when followed by a `^Z` character, it suspends the `telnet` utility. The initial state is to disable the `rlogin` escape character.

`start` If the TELNET `TOGGLE-FLOW-CONTROL` option is enabled, this character is the terminal's `start` character. The initial value for the start character is the terminal's `start` character.

`stop` If the TELNET `TOGGLE-FLOW-CONTROL` option is enabled, this character is the terminal's `stop` character. The initial value for the stop character is the terminal's `stop` character.

`susp` If TELNET is in `localchars` mode, or `LINEMODE` is enabled, and the `suspend` character is typed, a TELNET SUSP sequence (see `send susp` previously) is sent to the remote host. The initial value for the suspend character is the terminal's `suspend` character.

`tracefile` If it is set to a `-` symbol, tracing information is written to standard output (the default). This is the file to which the output, caused by `netdata` or `option` tracing being TRUE, is written.

`worderase` If TELNET is operating in `LINEMODE` or line-by-line mode, this character is the terminal's `worderase` character. The initial value for the word erase character is the terminal's `worderase` character.

`?` This displays the legal `set` (`unset`) commands.

`slc` *state* The `slc` command (Set Local Characters) sets or changes the state of the special characters when the TELNET `LINEMODE` option was enabled. Special characters are characters that are mapped to TELNET command sequences (such as `ip` or `quit`) or line editing characters (such as `erase` and `kill`). By default, the local special characters are exported. Values for *state* are as follows:

`check` Verifies the current settings for the current special characters. The remote side is requested to send all of the current special character settings, and if any discrepancies exist with the local side, the local side switches to the remote value.

`export` Switches to the local defaults for the special characters. The local default characters are those on the local terminal at the time when TELNET was started.

`import` Switches to the remote defaults for the special characters. The remote default characters are those on the remote system at the time when the TELNET connection was established.

`?` Prints help information for the `slc` command.

`status` Shows the current status of TELNET. This includes the peer to which you are connected, as well as the current mode.

`toggle` *arguments...*
Toggles (between `TRUE` and `FALSE`) various flags that control how TELNET responds to events. These flags can be set explicitly to `TRUE` or `FALSE` by using the `set` and `unset` commands listed previously. You can specify more than one argument. You can interrogate the state of these flags by using the `display` command. Valid arguments are as follows:

`authdebug` Turns on debugging information for the authentication code.

`autoflush` If `autoflush` and `localchars` are both `TRUE`, when the `ao`, `intr`, or `quit` characters are recognized (and transformed into TELNET sequences; see the preceding `set` command for details), TELNET refuses to display any data on the user's terminal until the remote system acknowledges (through a TELNET `TIMING MARK` option) that it has processed those TELNET sequences. If the terminal user has not done an `stty noflsh`, the initial value for this toggle is `TRUE`; otherwise they are `FALSE` (see `stty`(1)).

`autoencrypt`
`autodecrypt`
When the TELNET `ENCRYPT` option is negotiated, by default the actual encryption (decryption) of the data stream does not start automatically. The `autoencrypt` (`autodecrypt`) command states that encryption of the output (input) stream should be enabled as soon as possible.
Note: Because of export controls, Cray Research does not support the TELNET `ENCRYPT` option outside the United States and Canada.

`autologin` If the remote side supports the TELNET `AUTHENTICATION` option, TELNET attempts to use it to perform automatic authentication. If the `AUTHENTICATION` option is not supported, the user's login name is propagated through the TELNET `ENVIRON` option. This command is the same as specifying the `-a` option on the `open` command.

`autosynch` If `autosynch` and `localchars` are both TRUE, when either the `intr` or `quit` character is typed (see `set` listed previously for descriptions of the `intr` and `quit` characters), the resulting TELNET sequence sent is followed by the TELNET SYNCH sequence. This procedure should cause the remote system to begin deleting all previously typed input until both of the TELNET sequences are read and acted upon. The initial value of this toggle is `FALSE`.

`binary` Enables or disables the TELNET `BINARY` option for both input and output.

`inbinary` Enables or disables the TELNET `BINARY` option on input.

`outbinary` Enables or disables the TELNET `BINARY` option on output.

`crlf` If `crlf` is `TRUE`, carriage returns are sent as `<CR><LF>`. If `crlf` is `FALSE`, carriage returns are sent as `<CR><NUL>`. The initial value for this toggle is `FALSE`.

`crmod` Toggles carriage return mode. When you enable this mode, most carriage return characters received from the remote host are mapped into a carriage return followed by a line feed. This mode does not affect those characters that the user types; it affects only those received from the remote host. This mode is not very useful, unless the remote host sends only a carriage return, but never a line feed. The initial value for this toggle is `FALSE`.

`debug` Toggles socket-level debugging (useful only to the super user). The initial value for this toggle is `FALSE`.

`encdebug` Turns on debugging information for the encryption code.

`localchars`
If this value is TRUE, the `flush`, `interrupt`, `quit`, `erase`, and `kill` characters (see `set` listed previously) are recognized locally, and they are transformed into appropriate TELNET control sequences (respectively `ao`, `ip`, `brk`, `ec`, and `el`; see `send` listed previously). The initial value for this toggle is `TRUE` in line-by-line mode, and `FALSE` in character-at-a-time mode. When you enable the `LINEMODE` option, the value of `localchars` is ignored, and it is assumed to be always `TRUE`. If `LINEMODE` was ever enabled, `quit` is sent as `abort`, and `eof` and `suspend` are sent as `eof` and `susp` (see `send` listed previously).

`netdata` Toggles the display of all network data (in hexadecimal format). The initial value for this toggle is `FALSE`.

`options` Toggles the display of some of the internal TELNET protocol processing (related to TELNET options). The initial value for this toggle is `FALSE`.

`prettydump`
When you enable the `netdata` toggle, and `prettydump` is enabled, the output from the `netdata` command is formatted in a more user-readable format. Spaces are inserted between each character in the output, and the beginning of any TELNET escape sequence is preceded by a `*` symbol to aid in locating them.

`skiprc` When the `skiprc` toggle is `TRUE`, TELNET skips the reading of the `.telnetrc` file in the user's home directory when connections are opened. The initial value for this toggle is `FALSE`.

`termdata` Toggles the display of all terminal data (in hexadecimal format). The initial value for this toggle is `FALSE`.

`verbose_encrypt`
When the `verbose_encrypt` toggle is `TRUE`, TELNET prints out a message each time encryption is enabled or disabled. The initial value for this toggle is `FALSE`.
Note: Because of export controls, Cray Research does not support data encryption outside the United States and Canada.

`?` Displays the legal `toggle` commands.

`z` Suspends `telnet`. This command works only when the user is using the `csh`(1) command.

`![`*command*`]`
Executes a single command in a subshell on the local system. If you omit *command*, an interactive subshell is invoked.

`?[`*command*`]`
Gets help. With no arguments, TELNET prints a help summary. If you specify a command, TELNET prints the help information for only that command.

NOTES

On some remote systems, you must turn off echo manually when in line-by-line mode.

In line-by-line mode or `LINEMODE`, the terminal's `eof` character is recognized (and sent to the remote system) only when it is the first character on a line.

If this utility is installed with a privilege assignment list (PAL), a user who is assigned the following privilege text upon execution of this command is allowed to perform the action shown:

Privilege Text	Action
`admin`	Allowed to use `telnet` over a restricted interface.

If this utility is installed with a PAL, a user with one of the following active categories is allowed to perform the action shown:

Active Category	Action
`system`, `secadm`, `sysadm`	Allowed to use `telnet` over a restricted interface.

If the `PRIV_SU` configuration option is enabled, the super user is allowed to use `telnet` over a restricted interface.

ENVIRONMENT VARIABLES

TELNET uses at least the `HOME`, `SHELL`, `DISPLAY`, and `TERM` environment variables. To propagate other environment variables to the other side, use the TELNET `ENVIRON` option.

FILES

`~/.telnetrc`	User's `telnet` directory

SEE ALSO

`csh`(1), `rlogin`(1B), `rsh`(1), `stty`(1)

`inet`(3C), `krb_realmofhost`(3K) in the *UNICOS System Libraries Reference Manual*, Cray Research publication SR–2080

`hosts`(5) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

`kerberos`(7) (available only online)

Software Overview for Users, Cray Research publication SG–2052

NAME

`test` – Performs a conditional evaluation

SYNOPSIS

`test` *expr*
`[` *expr* `]`

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4
AT&T extensions (`-a`, `-G`, `-h`, `-k`, `-L`, `-O`, and `-S` primaries)
CRI extensions (`-m` and `-M` primaries)

DESCRIPTION

The `test` utility evaluates expression *expr* and, if its value is true, returns an exit status of 0 (true); otherwise, a nonzero (false) exit status is returned. Without arguments, `test` also returns a nonzero exit status.

You must specify all operators and elements of primaries as separate arguments.

The following primaries are used to construct *expr*:

`-b` *file* True if *file* exists and is a block special file.

`-c` *file* True if *file* exists and is a character special file.

`-d` *file* True if *file* exists and is a directory.

`-e` *file*
`-a` *file* True if *file* exists.

`-f` *file* True if *file* exists and is a regular file.

`-g` *file* True if *file* exists and its set-group-ID flag is set

`-G` *file* True if *file* exists and its group matches the effective group of this process.

`-h` *file* True if *file* exists and is a symbolic link.

`-k` *file* True if *file* exists and its sticky bit is set. This is always false; there is no sticky bit in UNICOS.

`-m` *file* True if *file* exists and is a migrated file (type IFOFL).

`-M` *file* True if *file* exists and is a migrated file (has a DMF handle).

`-n` *string* True if the length of the string *string* is nonzero.

`-O` *file* True if *file* exists and is owned by the effective user ID of this process.

`-p` *file* True if *file* exists and is a fifo (named pipe) special file.

`-s` *file* True if *file* exists and has a size greater than 0.

`-r` *file* True if *file* exists and can be read.

`-t` *fildes* True if the open file whose file descriptor number is *fildes* (1 by default) is associated with a terminal device.

`-u` *file* True if *file* exists and its set-user-ID flag is set.

`-w` *file* True if *file* exists and can be written. True indicates that only the write flag is on.

`-x` *file* True if *file* exists and can be executed. True indicates that only the execute flag is on. If *file* is a directory, true indicates that *file* can be searched.

`-z` *string* True if the length of string *string* is 0.

string True if *string* is not the null string.

s1 `=` *s2* True if strings *s1* and *s2* are identical.

s1 `!=` *s2* True if strings *s1* and *s2* are *not* identical.

n1 `-eq` *n2* True if the integers *n1* and *n2* are algebraically equal.

n1 `-ne` *n2* True if the integers *n1* and *n2* are algebraically not equal.

n1 `-gt` *n2* True if the integer *n1* is algebraically greater than the integer *n2*.

n1 `-ge` *n2* True if the integer *n1* is algebraically greater than or equal to the integer *n2*.

n1 `-lt` *n2* True if the integer *n1* is algebraically less than the integer *n2*.

n1 `-le` *n2* True if the integer *n1* is algebraically less than or equal to the integer *n2*.

You may combine these primaries with the following operators:

`!` Unary negation operator.

`-a` Binary AND operator.

`-o` Binary OR operator (`-a` has higher precedence than `-o`).

`(` *expr* `)` Parentheses for grouping.

NOTES

The `test` utility is a built-in utility to the standard shell (`sh`(1)). An executable version of `test` is available in `/usr/bin/test`.

Because parentheses are meaningful to the shell, they must be escaped.

CAUTIONS

In the second form of the utility (that is, the one that uses `[]` rather than the word `test`(1)), you must delimit the brackets by blanks.

EXIT STATUS

The `test` utility exits with one of the following values:

0 *expression* evaluated to true.

1 *expression* evaluated to false or *expression* was missing.

>1 An error occurred.

SEE ALSO

`find`(1), `sh`(1)

NAME

`tftp` – Invokes the trivial file transfer program

SYNOPSIS

`/usr/ucb/tftp` [`-S` *tos*] [*host*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `tftp` utility is the user interface to the Internet TFTP (Trivial File Transfer Protocol), which allows users to transfer files to and from a remote machine. (See the `connect` command that follows.)

The `tftp` utility accepts the following options:

`-S` *tos* Sets the IP Type-of-Service (TOS) option for the connection to the value *tos*, which may be a numeric TOS value or a symbolic TOS name found in the `/etc/iptos` file.

host Specifies the network name for the host computer.

After `tftp` is running, it issues the `tftp>` prompt and recognizes the following commands:

`connect` *host-name* [*port*]
Sets up *host-name* (and optionally *port*) for transfers. The TFTP protocol, unlike the FTP protocol, does not maintain connections between transfers; thus, the `connect` command does not actually create a connection, but merely remembers the host that is to be used for transfers. You do not have to use the `connect` command; you can specify the remote host as part of the `get` or `put` command.

`mode` *transfer-mode*
Sets up the mode for transfers; *transfer-mode* can be `netascii` or `octet`. The default is `netascii`. The TFTP protocol defines `netascii` and `octet`. You can use `ascii` as an alias for `netascii`; you can use `binary` and `image` as aliases for `octet`. If *transfer-mode* is not specified, the current mode is printed.

`put` *file*
`put` *localfile remotefile*
`put` *file1 file2 ... fileN remote-directory*
Writes a file or set of files to the specified remote file or directory. The destination can be in one of two forms: a file name on the remote host, if the host has already been specified, or a string of the form *host:filename* to specify both a host and file name at the same time. If the latter form is used, the specified host name becomes the default for future transfers. If the *remote-directory* form is used, you must specify at least three source files.

`get` *filename*
`get` *remotename localname*

`get` *file1 file2 . . . fileN*
: Gets a file or set of files from the specified remote file(s), which can be in one of two forms: a file name on the remote host, if the host has already been specified, or a string of the form *host:filename* to specify both a host and file name at the same time. If the latter form is used, the last host name specified becomes the default for future transfers.

`aput` *file*
`aput` *localfile remotefile*
`aput` *file1 file2 . . . fileN remote-directory*
: Uses the Kerberos-authenticated send file to write a file or set of files to the specified remote file or directory. The destination can be in one of two forms: a file name on the remote host, if the host has already been specified, or a string of the form *host:filename* to specify both a host and file name at the same time. If the latter form is used, the specified host name becomes the default for future transfers. If the *remote-directory* form is used, you must specify at least three source files.

`aget` *filename*
`aget` *remotename localname*
`aget` *file1 file2 . . . fileN*
: Uses a Kerberos-authenticated receive file to get a file or set of files from the specified remote file(s), which can be in one of two forms: a file name on the remote host, if the host has already been specified, or a string of the form *host:filename* to specify both a host and file name at the same time. If the latter form is used, the last host name specified becomes the default for future transfers.

`quit` Exits `tftp`. An EOF command also exits.

`verbose`
: Toggles verbose mode.

`trace` Toggles packet tracing.

`status` Shows current status.

`rexmt` *retransmission-timeout*
: Sets the per-packet retransmission time-out (in seconds).

`timeout` *total-transmission-timeout*
: Sets the total transmission time-out (in seconds).

`ascii` Specifies shorthand for `mode ascii`.

`binary` Specifies shorthand for `mode binary`.

`?` [*command-name . . .*]
: Prints help information for the indicated command.

SEE ALSO

`hosts`(5) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

NAME

`ticksum` – Provides a time independent checksum of a file

SYNOPSIS

`ticksum` *file*
`ticksum` *file1 file2*

IMPLEMENTATION

All Cray Research systems

STANDARDS

CRI extension

DESCRIPTION

The *time independent* checksum utility, if invoked with one argument, calculates a checksum of the named file and writes it (in hexadecimal) to standard output. For the specific kinds of binary files it recognizes, information such as the time of compilation and the version of the generating product is ignored in calculating the checksum.

If invoked with two arguments, the two files are compared by comparing their checksums calculated as above. A zero exit status indicates that the checksums were the same; a non-zero exit status that the checksums differed.

EXIT STATUS

The `ticksum` utility exits with one of the following values:

0 Checksum(s) computed; if two files given, the checksums were identical.

>0 Files differ, by the checksum criterion.

SEE ALSO

`cksum`(1)

NAME

`time` – Times a simple command

SYNOPSIS

`time` [`-p`] *utility* [*argument* ...]

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4
CRI extensions (output format)

DESCRIPTION

The *utility* with any *arguments* is executed; after it is complete, `time` writes the elapsed time during the utility, the time spent in the system, and the time spent in execution of the utility to standard error. By default, times are reported in seconds and in the corresponding number of CPU clock cycles.

`time` accepts the following options and arguments:

`-p`	Times are reported only in seconds in a standard and portable format.
utility	Name of the utility to be timed.
argument ...	Arguments specific to the utility being timed.

NOTES

The `csh`(1) utility has a built-in `time` utility with slightly different characteristics. See `csh`(1).

EXIT STATUS

The `time` utility exits with a 0 status if the timed *utility* terminates because of a signal. If the utility specified by *utility* could not be found, the exit status is 127. If the utility was found, but cannot be invoked, the exit status is 126. If some other error occurs within the `time` utility, the exit status is 2. Otherwise, the exit status shall be that of the timed *utility*.

SEE ALSO

`csh`(1), `ja`(1), `sar`(1), `timex`(1)

`times`(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

NAME

timex – Times a command and reports process data and system activity

SYNOPSIS

timex [-f] [-h] [-k] [-m] [-o] [-p] [-r] [-s] [-t] *command* [*args*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

When you invoke the timex utility, the given *command* is executed; the elapsed time, user time, and system time spent in execution are reported in seconds. Optionally, process accounting data for the *command* and all of its children can be listed or summarized, and total system activity during the execution interval can be reported.

The output of timex is written on standard error.

The timex utility accepts the following options:

-f Prints the fork/exec flag and system exit status columns in the output. This option applies when -p is specified; otherwise, it is ignored.

-h Displays the fraction of total available CPU time consumed by the process during its execution. This "hog factor" is computed as follows:

(*total CPU time*)/(*elapsed time*)

This option applies when -p is specified; otherwise, it is ignored.

-k Displays total kcore-minutes; this is an integral of memory usage over time. One kcore minute is 1024 words used for 1 minute. This option applies when -p is specified; otherwise, it is ignored.

-m Shows mean core size. This option applies when -p is specified; otherwise, it is ignored.

-o Reports the total number of blocks read or written and total characters transferred by *command* and all its children.

-p Lists process accounting records for *command* and all its children. The number of blocks read or written and the number of characters transferred are always reported.

-r Shows CPU factor (*user time*)/(*system-time* + *user-time*). This option applies when -p is specified; otherwise, it is ignored.

-s Reports total system activity (not just that due to *command*) that occurred during the execution interval of *command*. All data items listed in sar(1) are reported.

-t Shows separate system and user CPU times. This option applies when -p is specified; otherwise, it is ignored.

args Arguments for the command being timed.

WARNINGS

Process records associated with *command* are selected from the `/usr/adm/acct/day/pacct` accounting file by inference, because process genealogy is not available. Background processes that have the same user ID, terminal ID, and execution are spuriously included.

EXAMPLES

Example 1: A simple example follows:

```
timex -ops sleep 60
```

Example 2: A terminal session of arbitrary complexity can be measured by timing a subshell, as follows:

```
$ timex -opskmt sh
$ <sub-shell session commands>
   .
   .
   .
$ exit
.$
```

SEE ALSO

`acctcom`(1), `ja`(1), `sar`(1), `time`(1)

NAME

tmpdir – Creates a unique temporary directory

SYNOPSIS

tmpdir *path*

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The tmpdir utility creates a unique directory in *path*. It is useful when creating secondary job temporary directories on alternate devices. Before creating a directory, tmpdir checks the /etc/tmpdir.users file for authorization. Before exiting normally, tmpdir prints the name of the temporary directory to stdout.

The tmpdir utility accepts the following argument:

path Location of the temporary directory to be created.

Either init(8) or the Network Queuing System (NQS) will delete the files in temporary directories at job end.

NOTES

If tmpdir is executed in a secure system, the temporary directories can be found in the files ${TMPDIR}/.tmpdir[a-z]. The suffix a is added when the first tmpdir is executed. Subsequent executions of tmpdir may not be able to access .tmpdira for write if the user has raised either his security level or compartment settings. Thus, tmpdir will try to open .tmpdirb, .tmpdirc, and so on, looking for a file that it can create. The user can raise the security level and compartment setting a total of 26 times with intervening tmpdir executions. All tmpdir executions at the same security level and compartments will write to the same file.

MESSAGES

tmpdir: TMPDIR environment variable not set
: Either init or NQS did not set TMPDIR, or you unset it before executing tmpdir.

tmpdir: *user* is not in users file
: You are not authorized to use tmpdir.

tmpdir: *path* is not an authorized path
: You are not authorized to create directories in *path*.

tmpdir: too many tmpdir files
: You have changed security levels or compartments and executed tmpdir more than 26 times.

`tmpdir: file not owned by root:` *file*
The `tmpdir` file *file* is not owned by `root`, so it is not a valid `tmpdir` file.

BUGS

Changing the `TMPDIR` environment variable prevents cleanup by NQS or `init`(8).

CAUTIONS

If the `-` argument is used, the `TMPDIR` environment variable for the specified user is set to `JTMPDIR`, which is defined in `/usr/include/tmpdir.h`.

EXAMPLES

The following shell script fragment is used to create a temporary directory:

```
.
.
.
#
# shell script fragment to create a unique temporary directory
#
TMP2=‘tmpdir /ram‘
if [ $? != 0 ]
then
        echo "tmpdir failed" >&2
        exit 1
fi
#
# alias unit 11 an actual filename on /ram
#
assign -a ${TMP2}/fort.11 u:11
.
.
.
```

FILES

`${TMPDIR}/.tmpdir`	List of temporary directories created by `tmpdir`.
`/etc/tmpdir.users`	Authorization file for `tmpdir`.

SEE ALSO

`chown`(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

`tmpdir.users`(5) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

`cleantmp`(8), `init`(8), in the *UNICOS Administrator Commands Reference Manual*, Cray Research publication SR–2022

NAME

`touch` – Updates access and modification times of a file

SYNOPSIS

`touch` [`-a`] [`-c`] [`-m`] [`-r` *ref_file*] *files*
`touch` [`-a`] [`-c`] [`-m`] [`-t` *time*] *files*

Obsolescent version; may not be supported in future releases:

`touch` [`-a`] [`-c`] [`-m`] [*date_time*] *files*

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The `touch` utility updates the access and modification times of each file.

You may specify the time used by the `-t` *time* argument, the corresponding time field(s) of the file referenced by the `-r` *ref_file* argument, or the *date_time* operand. If you omit all of these options, `touch` uses the current time. The file name is created if it does not exist. If you omit all options, `touch` updates the access times and modification times.

The `touch` utility accepts the following options and operands:

`-a` Causes `touch` to update only the access times. The modification time is not changed unless you also specify `-m`.

`-c` Silently prevents `touch` from creating the file if it did not previously exist.

`-m` Causes `touch` to update only the modification times. The access time is not changed unless you also specify `-a`.

`-r` *ref_file* Uses the corresponding time of the file specified by the path name *ref_file*, rather than the current time.

`-t` *time* Uses the specified *time* instead of the current time. The argument is of the form:

[[*CC*]*YY*]*MMDDhhmm*[*.SS*]

Each two digits represents the following:

CC The first two digits of the year (the century).
YY Second two digits of the year.
MM Month of the year (01–12).
DD Day of the month (01–31).

hh Hour of the day (00–23).
mm Minute of the hour (00–59).
SS Second of the minute (00–61).

If neither *CC* or *YY* is given, the current year is assumed. If *YY* is specified but *CC* is not, *CC* will become 19 if *YY* is in the range 69–99. *CC* becomes 20 if *YY* is in the range 00–68.

files Specifies each file path name whose times are to be modified.

date_time Uses the specified *date_time* rather than the current time. The operand is of the form:

MMDDhhmm[*yy*]

MM, *DD*, *hh*, and *mm* are as described for the *time* argument to the `-t` option, and the optional *yy* is interpreted as follows:

- If not specified, the current year is used.
- If *yy* is in the range 69–99, the year 1969–1999 is used.
- If *yy* is in the 00–68, the year 2000–2068 is used.

If you omit the `-r` option, the `-t` option is not specified, at least two operands are specified, and the first operand is an 8- or 10-digit decimal number, the first operand is assumed to be a *date_time* operand; otherwise, the first operand is assumed to be a *file* operand.

Only an appropriately authorized user can update time information for a file owned by another user.

NOTES

The `touch` utility updates the date and time displayed by the `ls`(1) utility.

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
`system`, `secadm`	Update time information of any file.
`sysadm`	Update time information of any file, subject to security label restrictions. Shell-redirected I/O is subject to security label restrictions.

If the `PRIV_SU` configuration option is enabled, the super user is allowed to update time information of any file.

EXIT STATUS

The exit status of `touch` is the number of files for which the times could not be successfully modified (including files that did not exist and were not created).

SEE ALSO

`date`(1), `ls`(1)

`utime`(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

General UNICOS System Administration, Cray Research publication SG–2301

NAME

tpcatalog – Catalogs, recatalogs, or deletes a dataset in a front-end catalog

SYNOPSIS

tpcatalog [-d] *path*

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The tpcatalog command catalogs, recatalogs, or deletes a dataset in a front-end catalog.

The tpcatalog command accepts the following option and argument:

-d Deletes dataset from the catalog.

path Specifies the path name, a positional parameter, previously specified on the -P or -p option of the tpmnt(1) utility. By default, a new dataset is cataloged; however, if the dataset already exists in the catalog, it is recataloged.

The tpcatalog command returns an error if the specified *path* has not been accessed yet, or if the *path* is still open. Also, the tpcatalog command returns an error if front-end servicing is not turned on in the tape daemon.

NOTES

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
secadm, sysadm	Allowed to use this command.

If the PRIV_SU configuration option is enabled, the super user is allowed to use this command.

EXIT STATUS

If tpcatalog completes successfully, 0 is returned; otherwise, a nonzero value is returned. Where possible, this exit status code is normalized to the last three digits. Exit status values are documented in the *Tape Subsystem User's Guide*, Cray Research publication SG–2051.

FILES

/usr/include/taperr.h Tape daemon error codes

SEE ALSO

`tpmnt`(1)

`tpdaemon`(8), `tpset`(8) in the *UNICOS Administrator Commands Reference Manual*, Cray Research publication SR–2022

Tape Subsystem User's Guide, Cray Research publication SG–2051

NAME

tplist – Lists contents of tape volume

SYNOPSIS

tplist [-b *buffer_size*] [-B] [-c *copy_file*] [-C *copy_file*] [-e *tm_count*] [-g *resource_name*] [-G *resource_name*] [-k] [-n *file_count*] [-p] [-P] [-r] [-R] [-s *skip_count*] [-u] [-v *vilist*] [-V *vilist*] *pathname*

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The tplist utility displays the contents of one or more tape volumes. The data displayed is selected information from the volume header record, the header 1 record, and the header 2 record, along with the number of records in the data sections.

The tplist program automatically issues rsv(1), tpmnt(1), and rls(1) utilities to access a tape volume. If the tape daemon did not create the files, the first end-of-volume will cause tplist to exit.

The tplist utility accepts the following options:

-b *buffer_size* Sets the read/write buffer size (in bytes) to the specified value. The defaults are as follows:

Computer System	Default
CRAY J90 series	48 kilobytes or 49152 bytes
All other Cray Research systems	128 kilobytes or 131072 bytes

If the default buffer size or the value specified is a value smaller than the physical record size, it is too small to identify the contents of the tape and you will receive an error. If the buffer is too small, increase the buffer size and rerun the command until the command executes and the buffer size is determined.

-B Selects blp as the label option for the internal tpmnt(1) command. For additional information on label options, see the tpmnt(1) man page.

-c *copy_file* Copies the input file to the path *copy_file*.

-C *copy_file* Copies the input file to the path *copy_file* and converts from EBCDIC to ASCII. The -c and the -C options are mutually exclusive.

-e *tm_count* Sets the end-of-volume tape mark count to *tm_count*. The default is 2.

-g *resource_name* Sets the device resource group to *resource_name*.

-G *resource_name* Sets the device resource group to *resource_name* for copy volumes.

`-k`	Allows you to verify the output tapes that result from a `tplist` copy operation after the operation has successfully completed.
`-n` *file_count*	Specifies the number of files to copy. The default is 1.
`-p`	Allows you to specify additional parameters for the primary `tpmnt`(1) utility. Double quotation marks must enclose the parameter list.
`-P`	Allows you to specify additional parameters for the copy or secondary `tpmnt`(1) utility. Double quotation marks must enclose the parameter list.
`-r`	Displays unformatted labels.
`-R`	Retains the volume serial number (VSN) on the output tape if labeled tape copy. If you omit the `-R` option, the VSN from the input tape (as specified by the `-v` option) will be written to the output tape (as specified by the `-V` option).
`-s` *skip_count*	Skips *skip_count* files before listing or copying.
`-u`	Sets the no unload flag to keep the primary volume mounted when `tplist` completes.
`-v` *vilist*	Specifies volume identifier list for one or more input tapes. Same format as for `tpmnt`(1).
`-V` *vilist*	Specifies volume identifier list for one or more tapes for a copy operation. Same format as for `tpmnt`(1).
pathname	Specifies the path name to use for input.

NOTES

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	**Action**
`secadm`	Allowed to use this command.

If the `PRIV_SU` configuration option is enabled, the super user is allowed to use this command.

To access Data Migration Facility (DMF) tapes, the user must also have the active category of `datamgr`.

MESSAGES

Messages that originate with the tape daemon or any of its service programs are put in the tape message file (`tape.msg`). Other messages may result from I/O errors or tape system failures, and they are printed on `stderr`.

EXAMPLES

The following examples illustrate different uses of the `tplist` utility.

Example 1: The `tplist` utility lists the contents of the tapes `U00600` and `U00601`:

```
tplist -v u00600:u00601 pathname
```

Example 2: The `tplist` utility lists the contents of a round tape `scrsl`:

```
tplist -g TAPE -v scrsl pathname
```

Example 3: The `tplist` utility copies the first 20 files from `u00600` to `mycopy`:

```
tplist -v u00600 -c y -V mycopy -R -n 20 x
```

FILES

`/usr/include/taperr.h`	Tape daemon error codes

SEE ALSO

`rls`(1), `rsv`(1), `tpmnt`(1)

Tape Subsystem User's Guide, Cray Research publication SG–2051

General UNICOS System Administration, Cray Research publication SG–2301

NAME

tpmnt – Requests a tape mount for a tape file

SYNOPSIS

tpmnt [-a] [-b *block-size*] [-B] [-c *path-name*] [-C *catop1*[:*catop2*]] [-d *density*] [-D *device-name*]
[-f *file-identifier*] [-F *record-format*] [-g *device-group-name*] [-h *file-access-mode*]
[-H *volume-access-mode*] [-i on | off] [-I] [-j *CRL-volume-ID*] [-J *CRL-volume-ID-file*]
[-k *file-password*] [-K *volume-password*] [-l *label-type*] [-L *record-length*] [-m *front-end id*] [-M]
[-n] [-o] [-O *offset-of-first-volume-id*] [-p *path-name*] [-P *path-name*] [-q *file-sequence-number*]
[-Q *file-sequence-number*] [-r *ring-option*] [-R *path-name*] [-s] [-S *volume-set-name*]
[-t *retention-period*] [-T] [-u] [-U] [-v *ivid*[=*evid*][=*fid*][/*part*][:*ivid*[=*evid*][=*fid*][/*part*]...]
[-V *volume-identifier-file*] [-w] [-W *pool-name*] [-x *expiration-date*] [-X]
[-y *volume-set-expiration-date*] [-z]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The tpmnt utility requests the system operator to mount the specified tape volume and defines the characteristics of a tape file for the tape subsystem.

The -h, -H, -j, -J, -k, -K, -S, -W, -X, and -y options are valid only when the Cray/REELlibrarian (CRL) is in use. These options are documented at the end of the options list.

The tpmnt utility accepts the following options:

-a
Indicates that data is added to the end of an existing file. If the file does not exist, a new one is created. After the program opens the file, the data is positioned after the last block of the file. You cannot use this option with the -o or -n option.

-b *block-size*
Specifies the maximum block length in bytes (represented in decimal). The default maximum block size is specified in the tape configuration. If you specify this option, the specified *block-size* is used, rather than the value in the label of an existing tape file. If you are creating a new file and you do not specify this option, an installation default is used. (The default is specified in the tape configuration file.) If you do not use this option when writing to an existing file, the value of the *block-size* field in the label is used. When you are reading or creating a tape file, you should specify the largest block to be written.

For tapes read on ER90 devices, this option specifies the maximum block size. The value can exceed the actual data block size. However, an error is returned on the I/O request if *block-size* is less than the actual data block size. For tapes created on ER90 devices, this option specifies the size of all blocks to be written to tape. If this option is not specified when creating a new tape, an installation default is used. If writing to an existing file, the existing block size is used. This option is not valid for byte-stream tape files.

`-B` ER90 volumes only. For output tapes, this specifies that a blocked file section be created. For input files, this option is not used. For output tapes, a byte-stream file section is created by default. For input tapes, read requests are issued based on the mode in which the volume was created.

`-c` *path-name* Specifies a concatenated tape file. A previously executed `tpmnt` utility that used the `-p` or `-P` option established tape file *path-name*. If you specify this option, tape files will be concatenated to *path-name*. To concatenate several tape files, use sequential `tpmnt` utility with identical values for the `-c` option. The order of the commands determines the order in which the files are mounted. To read concatenated tape files, the tape subsystem automatically switches between files. The `-c`, `-p`, and `-P` options are mutually exclusive.

`-C` *catop1*[:*catop2*] Specifies catalog options to be implemented for the tape file, specified in the *file-identifier* of the `-f` option, on a servicing front end.

The following catalog options are available with the `-C` option:

`SAVE` Catalogs or recatalogs file in the front-end catalog

`DEL` Deletes file from the front-end catalog

`NO` Performs no operation on the front-end catalog

If the tape access executes normally, the catalog option specified in *catop1* is implemented. If the tape access terminates abnormally, the catalog option specified in *catop2* is implemented. If you specify *catop1* alone, *catop2* defaults to `NO`. If you omit the `-C` option, no operation is performed on the front-end catalog.

`-d` *density* Specifies the density of the tape volume. Density may be 1600 b/i or 6250 b/i. You cannot use this option if IBM 3480-type drives are specified with the `-D` or `-g` option (`tpmnt` will abort).

`-D` *device-name* Requests a specific device name for the mount.

If the requested device is available, it will be used to mount all volumes of a job. If the requested device is busy (assigned to another user), the request will be queued until the device is available. If the requested device is not configured up, the mount request will fail. If the volume serial number (VSN) requested is premounted and idle on another drive, that drive will be used instead of the drive requested. When automatic volume recognition (AVR) is enabled, this option is not available.

`-f` *file-identifier* Identifies the file on the volume that was recorded or will be recorded in the HDR1 label. You can specify the file identifier in lowercase or uppercase letters; however, the system automatically converts lowercase letters to uppercase letters (ANSI standard requirement). If the result is longer than 17 characters, the last 17 characters are used. If, after any trailing blanks are removed, the result is less than 17 characters, it is padded with blanks. The system administrator configures whether the *file-identifier* is needed and checked. If you omit the *file-identifier* with this option, the file name portion of the path name specified in the `-p` or `-P` option is used as the *file-identifier*. This option is ignored for unlabeled tapes.

For CRL the format of the *file-identifier* may be the format of a CRL file name. See the `rlfedit`(1) command for information on CRL file name syntax.

`-F` *record-format* Specifies the value of the record format field in the HDR2, EOV2, and EOF2 labels of ANSI and standard IBM labels. Record format may be one of following:

`F` Fixed length (for both ANSI label and IBM standard label)

`D` Variable length with zoned decimal length indicator (for ANSI label)

`U` Undefined length (for both ANSI label and IBM standard label)

`V` Variable length (for standard IBM label)

This value is used when creating labels. The UNICOS operating system does not verify that the data format in the file agrees with this value, and there is no data conversion when the tape is written or read (see `assign`(1) for data conversion). If you omit the `-F` option when creating a new file, the default record format is `U`. If you omit the `-F` option when writing over an existing file, the record format in the label is used.

You may specify an optional second character. This character indicates the attributes of the data, and it may be one of the following:

`B` Blocked records

`S` Spanned or standard records

`R` Blocked and spanned or standard records

`-g` *device-group-name*

Specifies the name of the group to which the requested device belongs. If you omit *device-group-name*, it defaults to an installation-specified name. The device group name (*dgn*) field of `tpstat`(1) shows the allowable parameters.

`-i on | off` Specifies the status of the Improved Data Recording Capability (IDRC) feature when you are using IBM cartridge model 3490 or 3480 tape drives that have this feature.

If you specify the `-i` option, the tape subsystem activates or deactivates IDRC accordingly when the file specified on the `tpmnt` utility is opened and when new volumes are mounted if the file is a multivolume file. The tape subsystem does not change the IDRC setting when the tape is unloaded.

Status is one of the following:

`on` Activates IDRC. When data is read from the device, the control unit decompresses the data automatically.

`off` Deactivates IDRC. Data is not compressed.

If you omit the `-i` option, you do not have any control over IDRC usage.

If you specify the `-i` option with incompatible hardware, an error occurs when data is first written to the device.

`-I` Specifies that mount verification using the format ID should be bypassed. This option requires that the user have bypass label or tape manager permission. This option is valid only with ER90 volumes.

`-l` *label-type* Defaults to an installation-specified label type. The *label-type* may be one of the following:

`al` ANSI label

`blp` Bypass label processing

`nl` Not labeled

`sl` IBM standard label

`st` Single tape mark format (single tape mark terminates reel)

`ulp` User label processing. This argument causes the tape subsystem to ignore the label type (`unknown` is the operator mount message) and to rewind the tape to the beginning of the volume so the user can read the volume and header records. Normal access control is checked prior to allowing the user to read the tape labels. The tape is mounted with the write protect ring out to prevent any accidental writes at the beginning of the tape. This argument is used primarily by the `tplist`(1) utility to examine the contents of a tape, and it eliminates the need for `tplist`(1) to use `blp`.

Note: Special permission is needed for `blp` and `nl` labels. See your system administrator.

`-L` *record-length* Indicates the maximum record length in bytes (represented in decimal). If you specify this option, the specified *record-length* is used rather than the value in the HDR2, EOV2, and EOF2 labels of an existing tape file. If you are creating a new file, and you omit this option, an installation default value is used. If you do not specify this option when writing over an existing file, the value in the *record-length* field of the label is used. For nonlabeled tapes, this option is ignored.

`-m` *front-end id* Specifies a front-end ID. It identifies the servicing front end (32 character maximum).

`-M` Delays mount message until file is opened. A tape unit is assigned when you issue `tpmnt`.

`-n` Specifies that the file is a new file. You cannot use this option with the `-a` or `-o` option.

`-o` Specifies that the file is an old file. You cannot use this option with the `-a` or `-n` option.

`-O` *offset-of-first-volume-id*

Specifies an offset into the volume identifier list. The offset points to the volume that will be mounted first. The first volume identifier has an offset of 1, which is the default. The offset is set to 1 after the file is opened.

If you use the `-O` option to specify an offset to a volume identifier, the output of the `tpmql`(8) command is no longer accurate. The `tpmql`(8) command displays the entire list of volume identifiers. If an offset has been set, the list of identifiers from `tpmql`(8) may no longer correspond to the order in which the tapes have been accessed with the `tpmnt` utility.

`-p` *path-name* Specifies the path name of the tape file. This path name is used to create a file that will be associated with the device used by the tape file. If the file to which this path name points already exists, an error will be issued. You must specify a path name by using the `-p`, `-P`, or `-c` option; these options are mutually exclusive. Your program must use this path name for the open function. If the path name is not a full path name, `tpmnt` appends what you have specified to your working directory. You must not move or remove this file, because unpredictable results will occur. If you do not specify a *file-identifier* with the `-f` option, the file name portion of this path name is used as the *file-identifier*.

`-P` *path-name* Specifies the path name for this tape file. The file to which this path name points, if it exists, is removed before a new one is created. (No error will be issued.) You must specify a path name by using the `-p`, `-P`, or `-c` option. Your program must use this path name in the open function. If the path name is not a full path name, `tpmnt` appends what you have specified to your working directory. A file is created by using the path name. The file is associated with the device that the tape file will use. You must not move or remove this file, because unpredictable results will occur. If you do not specify a *file-identifier* with the `-f` option, the file name portion of the path name is used as the *file-identifier*.

`-q` *file-sequence-number*

Specifies the file sequence number of the file being processed. The default is 1, which indicates the first file on the tape. The following are valid values for *file-sequence-number*:

number Is used if *file-sequence-number* is specified as a number and the tape is either labeled or unlabeled.

If the tape is labeled, the file sequence number is compared to that recorded in the HDR1 label of the first file of an existing volume. the result determines how many files the tape subsystem must skip to position to the correct file.

If the tape is unlabeled, the tape subsystem skips *file-sequence-number* tape marks from the beginning of the volume to position to the correct file.

`u` Indicates that positioning should be based on a file name.

`n` Creates a new file at the end of the tape. The file sequence number is not validated until the file is opened.

`-Q` *file-sequence-number*

Skips to position `n-1` and verify the header and trailer label of the file before the desired file. This option is designed for users who are confident of the contents of single volume tapes. It reverts to `-q` processing if any of the following are true:

- CRL is in use.
- User tape marks are specified.
- Multivolume tape is being accessed.

`-r` *ring-option* Specifies whether the write protect ring should be in or out. The allowable value for *ring-option* is either `in` or `out`. If you omit `-r`, an installation-defined default is used.

`-R` *path-name* Lets you release a tape volume and request a new tape by using one command. Your can use `-R` in two ways. If `-R` is used to specify a path name specified by the `-p` or `-P` option of a previously executed `tpmnt` utility, a tape mounted on this path is remounted with a new volume. If `-R` is used in conjunction with the `-p` or `-P` option, the tape volume mounted on the path specified with `-R` is released, and the volume is mounted on the path specified with the `-p` or `-P` option.

`-s` Specifies that the file is to be protected on the servicing front end. The `-s` option is valid only for new files being cataloged on the servicing front end.

`-t` *retention-period* Specifies the number of days a labeled tape will be kept. The default retention period is 0. This option is ignored on nonlabeled tapes.

`-T` Lets you read or write tape marks that are embedded in data.

`-u` Disables tape unload at release time. Usually, a tape is unloaded automatically when a job terminates or releases the tape resources. This option is useful when a tape is used repeatedly, and it minimizes operator time spent mounting tapes. This option has the same effect as specifying the `-n` option on the `rls`(1) utility.

`-U` Specifies unbuffered I/O for transparent I/O. Data for an I/O operation should not be buffered. When `-U` is specified, the read size must be a multiple of 4096 bytes and must be greater than or equal to the maximum block size. If `-U` is specified on a write request, data is written to tape as a tape block. If `-U` is specified on a read request, a tape block is transferred into the user's read buffer.

`-v` *ivid*[=*evid*][=*fid*][/*part*]
[:*ivid*[=*evid*][=*fid*][/Vpart*]...*
Specifies the volume identifier of the tapes. This option specifies a list of volume identifiers. You can specify the volume identifiers in lowercase or uppercase letters; the system automatically converts the internal and external volume IDs to uppercase letters.
The *ivid* is the ID that appears on the volume label. The *evid* specifies the identifier that appears on the physical label of the tape reel; this differs from the volume ID on the volume label. If the external volume ID is not specified, the default is the internal ID. The *fid* specifies the volume identifier recorded on the tape during a volume format. If a format ID is not specified, the internal volume ID is used. The *fid* is used for mount verification (used for ER90 volumes only). The *part* specifies the partition number to which the volume should be positioned. The partition number must be in the range of 0 through 1023. The default is 0, the first partition on the volume (a partition number is valid only on ER90 volumes). The `-V` and `-v` options are mutually exclusive.

`-V` *volume-identifier-file*
Specifies the file that contains a list of volume identifiers and external volume identifiers that compose a multivolume tape file. The `-V` option is useful for tape files that have many volume identifiers. The `-V` and `-v` options are mutually exclusive.

`-w` Lets you suspend execution until the `tpmnt` request is completed. This lets you check status to determine whether the mount was successful.

`-x` *expiration-date* Specifies the Julian date (in *xyyddd* format) on which the tape file expires. The year (*yy*) in the date ranges from 00 to 99, and the days (*ddd*) range from 001 to 366. If *x* is a blank, the first two significant digits of the year are 19. If the first character is a 0, the first two significant digits of the year are 20.
If the first digit is @, using [`98`*xxx*] or [`99`*xxx*] overrides the special meaning that the years 1998 and 1999 have to the Catalog Management System (TMS) on an IBM Multiple Virtual Storage (MVS) Operating System. Entering the @ causes the real 1998 and 1999 expiration dates to be used.
For nonlabeled tapes, this option is ignored. If you use CRL, specify this option by using the CRL file expiration date syntax as that used on CRL file commands (`rlfedit`(1) and `rlfsubmit`(1)), as follows:

`Adays` The file expires if not accessed in a period longer than the number of days specified.

`Gnum` The file expires when the number of newer generations equals the number specified by `num`. Valid values for `num` are 0 through 99.

`I`	The file never expires.
`Rdays`	The file expires the specified number of days after it is created.
`S`	The file is always expired; scratch.
`Xdate`	The file expires on the specified date; the format of the date is *mm/dd/yy*.

`-z` ER90 volumes only. Specifies that the volume be left at its current position after a tape mount and before an unload. By default, the tape is positioned to the beginning of the tape. To use this option, users must have bypass label or tape manager permission.

CRL options:

`-h` *file-access-mode* Specifies CRL mode protection value for files. When creating or recataloging a tape file, use *file-access-mode* as the CRL mode protection value.

`-H` *volume-access-mode*
Specifies CRL mode protection value for volumes. When creating or recataloging a tape file, use *volume-access-mode* as the CRL mode protection value.

`-j` *CRL-volume-ID*[:*CRL-volume-ID*]
Specifies the use of *CRL-volume-ID* as the volume identifier. You can use this option instead of the `-v` or `-V` options to uniquely identify the list of volumes to use. When using CRL, you do have to specify the entire list of volume identifiers, because the list is retrieved in its entirety when the volume set has been identified by any option that identifies file name (`-f`), path name (`-p`), volume set name (`-S`), or volume identifier (`-v` or `-V`).

`-J` *CRL-volume-ID-file*
Specifies that this file contains a list of unique CRL volume IDs that constitute a multivolume tape file. You may use this option instead of the `-v` or `-V` option to uniquely identify the list of volumes to use. When using CRL, you do not have to specify the entire list of volume identifiers, because the list is retrieved in its entirety when the volume set has been identified by any option that identifies file name (`-f`), path name (`-p`), volume set name (`-S`), volume identifier (`-v` or `-V`), or first CRL volume identifier (`-j`).

`-k` *file-password* Specifies the password to be associated with a file that is being created, or the password to be compared with the password for an existing file.

`-K` *volume-password* Specifies the password to be associated with a volume that is being created, or the password to be compared with the password for an existing volume.

`-S` *volume-set-name* Specifies the CRL volume set name to be accessed. You can use this option in place of the `-v`, `-V`, `-j`, or `-J` option. All appropriate volume identifiers are retrieved from the CRL catalog for access.

`-W` *pool-name* Specifies the CRL pool name to be used when drafting volumes for use in writing new tape files. If you omit this option, CRL uses the default pool that has been assigned to the user.

`-X` Causes all file and volume activity to be processed by the CRL catalog server. If CRL is not enabled, specifying this option causes an error. If CRL is enabled and mandatory, this option is not required; if it is used, it has no effect.

`-y` *volume-set-expiration-date*

Specifies the volume set expiration date for newly created volume sets. The format of the expiration date is the same as that used on the CRL volume set commands (`rlvcreate`(1), `rlvedit`(1), and `rlvsubmit`(1)), as follows:

`Adays`	The volume expires if not accessed in a period longer than the number of days specified.
`Gnum`	The volume expires when the number of newer generations equals the number specified by `num`. Valid values for `num` are 0 through 99.
`I`	The volume never expires.
`L`	The volume expires when all files in the volume set have expired.
`Orot_name`	The volume set follows the `rot_name` rotation schedule.
`Rdays`	The volume expires the specified number of days after it is created.
`S`	The volume is always expired; scratch.
`Xdate`	The volume expires on the specified date; the format of the date is *mm/dd/yy*.
`Z`	Duplicate 98000 expiration date functionality. If you specify `Z` and the VSNs specified by the `-V` option are not found in the CRL catalog, CRL processing is bypassed and tape processing continues.

NOTES

The UNICOS system can enforce tape labeling. The tape daemon automatically sets the accessibility byte of the VOL1 and HDR1 labels, and it writes the security label in the HDR2 label for each classified file written to tape. ANSI and IBM labels are supported.

EXIT STATUS

If `tpmnt` completes successfully, 0 is returned; otherwise, a nonzero value is returned. Where possible, this exit status code is normalized to the last three digits. Exit status values are documented in the *Tape Subsystem User's Guide*, Cray Research publication SG–2051.

EXAMPLES

The following examples illustrate different uses of the `tpmnt` utility.

Example 1: The `tpmnt` utility requests that volume 000011 be mounted, and tape processing begins at partition 3. The format ID of the mounted tape must be ER011. If a file is created, it will be a blocked file that consists of blocks of size 1,199,832 bytes.

```
tpmnt -v 000011==ER011/3 -l sl -p x -n -g ER90 -B -b 1199832
```

Example 2: The `tpmnt` utility requests that volume 000011 be mounted, and tape processing begins at partition 0. Format ID verification will not be done. If a file is created, it will be a byte stream file. If another volume is needed, the tape will be positioned to partition 1 of the same volume without unloading volume 000011.

```
tpmnt -v 000011/0:000011/1:000011/2 -l sl -p x -n -g ER90 -I
```

Example 3: The `tpmnt` utility requests that volume 000012 be mounted, and the tape is left at its load position. After opening the `tapefile` file, the first request to the tape file must be a request to establish the logical position. This is done by issuing a `TPC_PABS ioctl` request.

```
tpmnt -v 000012 -l blp -p x -f tapefile -g ER90 -B -b 1048576 -z
```

FILES

`/usr/include/errno.h`	Error code header file

SEE ALSO

`rls`(1), `rsv`(1), `tprst`(1), `tpstat`(1)

`tplabel`(8), `tpmql`(8) in the *UNICOS Administrator Commands Reference Manual*, Cray Research publication SR–2022

Tape Subsystem Administration, Cray Research publication SG–2307

Tape Subsystem User's Guide, Cray Research publication SG–2051

NAME

tpquery – Queries autoloaders for VSN information

SYNOPSIS

```
tpquery [-l] [-m loader_name] [-t trace_file] -v vsns
tpquery [-l] [-m loader_name] [-t trace_file] -V vsn_list_file
```

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The tpquery command queries an autoloader for information concerning one or more volume serial numbers (VSNs). The returned information contains text that indicates whether the specified VSNs are within the domain of the queried autoloader.

The tpquery command accepts the following options:

`-l` Provides the following information for each specified VSN:

loader name: vsn: text on cartridge state: possible cartridge location

If you omit the `-l` option, no information concerning the specified cartridge(s) is provided.

`-m` *loader_name* Specifies the autoloaders that the command will query. Enter one of the following for *loader_name*:

name Specifies the name of an autoloader in the tape configuration file. If you specify an autoloader name, tpquery directs its query to the specified autoloader after it has verified that the name appears in the tape configuration file, that the autoloader is connected to a network, and that the autoloader is in an UP or MANUAL state.

all or ALL Specifies all autoloaders listed in the tape configuration file that are connected to a network and that are in an UP or MANUAL state. If one of these autoloaders contains all the specified VSNs, the remaining autoloaders are not queried. The default is all.

`-t` *trace_file* Specifies the name of a file to which tpquery writes trace information to while the command is being executed. If you omit the `-t` option, tpquery executes without writing trace information.

`-v` *vsns* Specifies one or more cartridge VSNs in the format that is documented for the tpmnt(1) command. The `-v` and `-V` options are mutually exclusive; one is required.

`-V` *vsn_list_file* Specifies the name of a file that contains a list of one or more VSNs. The `-v` and `-V` options are mutually exclusive; one is required.

EXIT STATUS

If `tpquery` completes successfully and has found all specified VSNs in one autoloader, it returns a zero status. If the command executes successfully, but does not find all specified VSNs in one autoloader, it returns a status with value `122: ETQRY - ERRBASE`.

If `tpquery` fails, an exit status code in the range of 1 through 255 is returned; where possible, the number is normalized to the last three digits. Exit status values are documented in the *Tape Subsystem User's Guide*, Cray Research publication SG–2051.

EXAMPLES

Example 1: This example shows the `tpquery` command being executed for VSN `xxxxxx`. Because the `-m` option is omitted, `tpquery` interrogates all of the autoloaders. The `-l` option is specified; therefore, all output is displayed.

```
# tpquery -v xxxxxx -l
stksun: XXXXXX: volume_not_in_loader.
wolfy:  XXXXXX: volume_not_in_loader.
ibm:    XXXXXX: volume_not_in_loader.
# echo $?
122
```

This output shows that VSN `xxxxxx` is not located in any of the online autoloaders. The status code returned to the calling program is `ETQRY` (`122`). The output also shows that lowercase alphabetic characters in the VSN are converted to uppercase alphabetic characters.

Example 2: This example shows the `tpquery` command being executed for VSN `S66580`. Because the `-m` option is specified, `tpquery` interrogates only the autoloader, called `wolfy`. It displays all of the output because the `-l` option is specified;

```
# tpquery -m wolfy -v S66580 -l
wolfy:  S66580: volume_in_loader: home:         0,3,7,3.
# echo $?
0
```

This output shows that VSN `S66580` is located in `wolfy` and that the VSN is located in its home position, which has the address `0,3,7,3`. The status code returned to the calling program is 0.

Example 3: This example shows the `tpquery` command being executed for VSN `003600`. The command interrogates all of the autoloaders because the `-m` option is omitted and displays all output because the `-l` option is specified,

```
# tpquery -l -v 003600
stksun: 003600: volume_not_in_loader.
wolfy:  003600: volume_not_in_loader.
ibm:    003600: volume_in_loader: home.
# echo $?
0
```

This output shows that VSN 003600 is located in the ibm autoloader and that this VSN is located in its home position, which is not further substantiated for IBM and EMASS autoloaders. The status code returned to the calling program is 0.

Example 4: This example shows the tpquery command being executed for VSN 003600. Neither the -m or -l option is specified; tpquery interrogates all of the autoloaders and returns 0 to the calling program. This status code indicates that VSN 003600 was found in one of the online autoloaders.

```
# tpquery -v 003600
# echo $?
0
```

SEE ALSO

tpmnt(1)

Tape Subsystem User's Guide, Cray Research publication SG–2051

NAME

tprst – Displays reserved tape status for current job ID

SYNOPSIS

```
tprst
```

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The tprst command displays the status of the tapes reserved for the current job ID. The following information is provided on standard output (stdout):

dev grp	Device group name.
w	User is waiting for a device. Indicated by an asterisk (*).
rsvd	Number of reserved devices.
used	Number of devices that have been used or are in use.
available	Number of devices available for reservation.

EXIT STATUS

If no error has occurred, tprst exits with a return code of 0. If an error has occurred, tprst exits with one of the following error codes (in decimal).

Symbol	Value	Description
ETSYS	23	Tape subsystem error
ETDCE	31	Cannot communicate with the tape subsystem

The symbols for these codes are located in the /usr/include/taperr.h file.

EXAMPLES

The following example shows output from tprst. The user has two TAPE devices and one CART device reserved and has used one of the two TAPE devices. The user is not waiting for any devices.

```
dev grp  w     rsvd     used available
TAPE             2        1    1
CART             1        0    1
```

SEE ALSO

`rls`(1), `rsv`(1), `tpmnt`(1), `tpstat`(1)

Tape Subsystem User's Guide, Cray Research publication SG–2051

NAME

tpstat – Displays current tape status

SYNOPSIS

tpstat [-a] [-l]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The tpstat utility returns the status of all tape devices.

The tpstat utility accepts the following options:

-a Outputs device status only for tape devices that have a status other than DOWN. If you omit this option, tpstat displays the device status for all tape devices.

-l Outputs the device status in long format. The long format has an added field length for the block count and outputs the format ID of mounted volumes.

The tpstat utility provides the following status:

userid Specifies user ID; blank if device is not assigned.

jobid Specifies job ID of the user if device is assigned.

dgn Specifies device group name.

a Specifies automatic volume recognition (AVR) for this device. If a plus sign (+) is displayed, AVR is turned on; if a minus sign (–) is displayed, AVR is turned off. If an at sign (@) is displayed, overcommitted mount requests are available; if an at sign (@) is not displayed, overcommitted mount requests are not available.

stat Specifies status of the device; status can be one of the following:

- assn Device is assigned to a user.
- cnfp Device is waiting for a configuration up command to complete.
- conn Device is temporarily not available for assignment.

  The conn status can result from the situations described in the following examples, but is not limited to these situations:

  Example 1: If a loader drive is available, but is not the most desirable drive, you see conn in the tpstat output for the drive currently owned by a tape daemon child process while the child process is selecting a more desirable drive.

Example 2: If a tape unit in an AVR system is up and a tpmnt(1) request is made for a tape volume currently not on a tape unit, the available tape unit is temporarily assigned to the requesting task and tpstat shows conn. When the tape is mounted on that tape unit, it is assigned for the requesting task to use and the tpstat output shows assn.

dldr Device is waiting for a configuration down command to complete.

down Device is unavailable.

free Device is being freed by an asynchronous process.

idle Device is available and not in use.

sdwn Device has been made unavailable by the system.

unav Device is unavailable.

wdwn Device is waiting to go to down state.

wsdn Device is waiting to go to sdwn state.

wsup Device is waiting to go to idle state.

If the device is configured as a no unload device (see tpconfig(8)), or if you requested no unload at release time by specifying the -u option of tpmnt(1), or by specifying the -n option of rls(1), a plus sign (+) is displayed next to the assn or idle status.

dvn Specifies device name.

bx Specifies minor device number of the tape device.

i or ion Specifies the following:

i Specifies hexadecimal cluster number.

ion Specifies the node.

rl
r = Ring status if assigned
i = Ring in
o = Ring out
l = Label status if assigned
a = ANSI label
b = Bypass label
n = Not labeled
s = IBM standard label

ivsn Specifies the internal volume identifier if assigned. If the tape is not yet mounted, an asterisk sign (*) appears before the vsn.

evsn Specifies the external volume identifier if assigned. This is the physical outer label of the tape volume. For more information, see the -v option of tpmnt(1).

`fvsn` Specifies the format identifier if assigned. The format identifier is the identifier that is recorded on the volume when the volume is formatted.

`blks` Specifies number of tape blocks read or written if assigned.

`NQSid` Specifies Network Queuing System (NQS) batch job ID if the assigned tape has been submitted through NQS. You may use this ID for `qstat`(1) or `qdel`(1).

EXIT STATUS

If `tpstat` completes successfully, 0 is returned; otherwise, a nonzero value is returned. Where possible, this exit status code is normalized to the last three digits. Exit status values are documented in the *Tape Subsystem User's Guide*, Cray Research publication SG–2051.

EXAMPLES

In the following example, devices `tape100` through `cart509` are configured as `down`. Devices `cart300` and `cart301` are not assigned to a job; therefore, they are `idle`. They have IBM-standard tapes `WE0022` and `WE0023` mounted on them.

User `hew` with job id `113` has `cart302` assigned as ring-in, ANSI-labeled, with an external volume identifier equal to `XXX` and an internal volume identifier equal to `ISCAL`. Nine blocks of data have been processed.

User `jas` with job ID `44` has been assigned `cart303`. It has been specified to be no unload (+), so that the tape will remain mounted after user `jas` issues an `rls`(1) utility. This tape has been assigned as ring-in, nonlabeled, volume identifier equal to `ISCNL`, and with 30 blocks of data processed. AVR is set to `on` for all tapes.

All tapes are attached to IOS 0.

```
    userid jobid dgn  a stat  dvn     bx i rl ivsn   evsn   blks NQSid
                 TAPE + down  tape100 02 0
                 TAPE + down  tape101 03 0
                 TAPE + down  tape102 04 0
                 TAPE + down  tape103 05 0
                 CART + down  cart508 06 0
                 CART + down  cart509 07 0
                 CART + idle  cart300 10 0 is WE0022 WE0022
                 CART + idle  cart301 09 0 is WE0023 WE0023
    hew    113   CART + assn  cart302 11 0 ia ISCAL  XXX    9
    jas    44    CART + assn+ cart303 08 0 in ISCNL  ISCNL  30
                 CART + down  cart400 12 0
                 CART + down  cart401 13 0
                 CART + down  cart402 14 0
                 CART + down  cart403 15 0
```

The following `tpstat` example is from a system with scalable I/O.

```
userid   jobid dgn     a stat dvn      bx ion  rl ivsn   evsn   blks  NQSid
               STK4890 - down s4890s0  02 0000
               STK4890 - down s4890s1  03 0000
               DLT4000 - down d4000s0  04 0000
               DLT4000 - down d4000s1  05 0000
               IBM3490E- idle 3490s0   06 0000
               IBM3490E- down 3490s1   07 0000
               DAT     + down h1533sX  08 0100
               STK9490 - down s9490s0  09 0100
               STK9490 - down s9490s1  10 0100
               STK9490 - down s9490s2  11 0100
               STK9490 - down s9490s3  12 0100
               IBM3590 - down 3590s0   13 0100
               IBM3590 - down 3590s1   14 0100
               STKSD3  - down ssd3_s0  15 0100
               STKSD3  - down ssd3_s1  16 0100
```

FILES

`/usr/include/tapereq.h`	Tape daemon interface definition file

SEE ALSO

`qdel`(1), `qstat`(1), `rls`(1), `rsv`(1), `tpmnt`(1), `tprst`(1)

`tpconfig`(8) in the *UNICOS Administrator Commands Reference Manual*, Cray Research publication SR–2022

Tape Subsystem User's Guide, Cray Research publication SG–2051

NAME

tput – Initializes a terminal or query terminfo database

SYNOPSIS

```
tput [-T type] capname [parms ...]
tput [-T type] clear
tput [-T type] init
tput [-T type] longname
tput [-T type] reset
```

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4
AT&T extensions (*capname* and longname operands)

DESCRIPTION

The tput utility uses the terminfo(5) database to make the values of terminal-dependent capabilities and information available to the shell (see sh(1)), to initialize or reset the terminal, or return the long name of the requested terminal type. tput outputs a string if the attribute (*cap*ability *name*) is of type string, or an integer if the attribute is of type integer. If the attribute is of type Boolean, tput simply sets the exit code (0 for true if the terminal has the capability; 1 for false if it does not) and produces no output. Before using a value returned on standard output, the user should test the exit code. (See the EXIT STATUS section.) For a complete list of capabilities and the *capname* associated with each, see terminfo(5).

The tput utility supports the following option and operands:

-T *type* Indicates the *type* of terminal. Normally this option is unnecessary, because the default is taken from the TERM environment variable. If -T is specified, the shell variables LINES and COLUMNS will not be referenced.

capname Indicates the attribute from the terminfo(5) database.

parms... If the attribute is a string that accepts parameters, the *parms* arguments will be instantiated into the string. An all-numeric argument is passed to the attribute as a number.

clear Clears the terminal screen.

longname If the terminfo(5) database is present and an entry for the user's terminal exists (see -T *type*), the long name of the terminal will be output. The long name is the last name in the first line of the terminal's description in the terminfo(5) database (see term(5)).

init If the terminfo(5) database is present and an entry for the user's terminal exists (see -T *type*) the following will occur:

1. Delays specified in the entry will be set into the tty driver.
2. If the hardware knows about tabs (as determined by the definition of the `ht`, `hts`, and `it` capabilities), the tty driver will not expand tabs.
3. If the `iprog` capability is not null, the specified program will be executed.
4. The `is1` string is sent if it exists, followed by the `is2` string, if it exists.
5. An attempt is made to initialize the tabstops of a terminal every 8 characters; if it does not work, you can fix it by using either steps 6 or 7.
6. If not null, the file specified by the *if* capability is copied to the terminal.
7. If not null, the `is3` string is sent.

`reset` Instead of putting out initialization strings, the terminal's reset strings `rs1`, `rs2`, `rs3`, and `rf`. will be output if present. If the reset strings are not present, but initialization strings are, the initialization strings will be output. Otherwise, `reset` acts the same as `init`.

EXIT STATUS

The `tput` utility exits with one of the following values:

0 The requested string was written successfully.

1 Unspecified.

2 Usage error.

3 No information is available about the specified terminal type.

4 The specified operand is invalid.

>4 An error occurred.

EXAMPLES

Example 1: The following example initializes the terminal according to the type of terminal in the `TERM` environmental variable. This command should be included in everyone's `.profile` after the `TERM` environmental variable has been exported, as illustrated on the `profile`(5) man page:

```
tput init
```

Example 2: The following example resets an AT&T 5620 terminal, overriding the type of terminal in the `TERM` environment variable:

```
tput -T5620 reset
```

Example 3: The following command sends the sequence to move the cursor to row 0, column 0 (the upper left corner of the screen, usually known as the "home" cursor position):

```
tput cup 0 0
```

Example 4: The following command echoes the clear-screen sequence for the current terminal:

```
tput clear
```

Example 5: The following command prints the number of columns for the current terminal:

```
tput cols
```

Example 6: The following command prints the number of columns for the 450 terminal:

```
tput -T450 cols
```

Example 7: The following command sets the standard shell variables `bold` to begin stand-out mode sequence, and `offbold`, to end stand-out mode sequence, for the current terminal. This might be followed by a prompt:

```
bold=‘tput smso‘
offbold=‘tput rmso‘
echo "${bold}Please type in your name: ${offbold}\c"
```

FILES

`/usr/lib/terminfo/?/*`	Compiled terminal description database
`/usr/include/curses.h`	`curses`(3) header file
`/usr/include/term.h`	`terminfo`(5) header file

SEE ALSO

`sh`(1) to invoke the standard command interpreter and command-level language
`stty`(1) to set the options for a terminal
`tabs`(1)

`curses`(3) to optimize terminal screens (available only online)

`profile`(5) for information on format of shell start-up file
`term`(5)
`terminfo`(5) for information on terminal capability database
in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

NAME

`tr` – Translates characters

SYNOPSIS

`tr` [`-c`] [`-s`] *string1 string2*
`tr -s` [`-c`] *string1*
`tr -d` [`-c`] *string1*
`tr -d -s` [`-c`] *string1 string2*

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The `tr` utility copies standard input to standard output with the substitution or deletion of selected characters.

Input characters found in *string1* are mapped into the corresponding characters of *string2*.

The `tr` utility supports the following options:

`-c` Complements the set of characters specified by *string1*.

`-s` Replaces instances of repeated characters with a one character.

`-d` Deletes all occurrences of input characters specified by *string1*.

string1
string2 Translation control strings. Each string represents a set of characters to be converted into an array of characters used for the translation.

The *string1* and *string2* operands define two arrays of characters. You can use the constructs in the following list to specify characters or single-character collating elements:

character Any character not described by one of the following conventions.

octal Octal sequences can be used to represent characters that have specific coded values. An octal sequence consists of a backslash followed by the longer sequence of one-, two-, or three-octal digit characters (01234567). The sequence causes the character whose encoding is represented by the one-, two-, or three-digit octal integer to be placed into the array.

character The following backslash-escape sequences may be specified:

`\\` backslash
`\a` <alert>
`\b` <backspace>

```
\f   <form-feed>
\n   <newline>
\r   <carriage-return>
\t   <tab>
\v   <vertical-tab>
```

c-*c* Represents the range of collating elements between the range endpoints, inclusive, as defined by the current setting of the LC_COLLATE locale category. The starting endpoint precedes the second endpoint in the current collation order. The characters or collating elements in the range are placed in the array in ascending collation sequence.

[:*class*:] Represents all characters that belong to the defined character class, as defined by the current setting of the LC_CTYPE locale category. The following character class names are accepted when specified in *string1*: alnum, alpha, blank, cntrl, digit, graph, lower, print, punct, space, upper, and xdigit.

When you specify both the -d and -s options, any of the character class names are accepted in *string2*; otherwise, only character class names lower or upper are valid in *string2* and then only if the corresponding character class (upper and lower, respectively) is specified in the same relative position in *string1*. Such a specification is interpreted as a request for case conversion.

[=*equiv*=] Represents all characters or collating elements that belong to the same equivalence class as *equiv*, as defined by the current setting of the LC_COLLATE locale category. An equivalence class expression is allowed only in *string1*, or in *string2*, when it is being used by the combined -d and -s options.

[*x***n*] Represents *n* repeated occurrences of the character *x*. Because this expression is used to map multiple characters to one, it is valid only when it occurs in *string2*. If you omit *n* or it is 0, it is interpreted to be large enough to extend the *string2*-based sequence to the length of the *string1*-based sequence. If *n* has a leading 0, the value is interpreted as octal; otherwise, the value is interpreted as decimal.

NOTES

Currently, UNICOS supports only the POSIX and C locales. These locales support only the ASCII character set.

The tr utility returns an error for tr -d *string1 string2* to report superfluous *string2* arguments. In UNICOS releases prior to 8.0, *string2* was simply ignored and processing occurred only on *string1*.

EXIT STATUS

The tr utility exits with one of the following values:

0 All input was processed successfully.

>0 An error occurred.

EXAMPLES

Example 1: The following example creates a list of all words in *file1*, one word per line, in *file2*; a word is taken to be a maximal string of alphabetic characters. The strings are quoted to protect the special characters from interpretation by the shell; 012 is the ASCII code for a <newline> character.

```
tr -cs "[A-Z][a-z]" "[\012*]" < file1 > file2
```

Example 2: The following example performs the same function as example 1:

```
tr -cs "[:alpha:]" "[\n*]" < file1 > file2
```

Example 3: The following example creates a file2 in which all alphabetic characters are uppercase:

```
tr "[:lower:]" "[:upper:]" < file1 > file2
```

SEE ALSO

ed(1), sh(1)

NAME

troff – Typesets or formats documents

SYNOPSIS

troff [-a] [-f] [-i] [-z] [-F*dir*] [-m*name*] [-n*N*] [-o*list*] [-r*aN*] [-s*N*] [-T*dest*] [-u*N*] [*filename*] ...

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The troff utility formats text in the *filename* argument. Input to troff is expected to consist of text interspersed with formatting requests and macros. If no *filename* argument is present, troff reads standard input. A hyphen as a *filename* argument indicates that standard input is to be read at that point in the list of input files; troff reads the files named ahead of the hyphen in the arguments list, then text from the standard input, and then text from the files named after the hyphen.

The following options may appear in any order, but they all must appear before the first *filename*.

-a Sends a printable approximation of the formatted output to the standard output file.

-f Does not print a trailer after the final page of output or causes the postprocessor to relinquish control of the device.

-i Reads the standard input after the input files are exhausted.

-z Suppresses formatted output. Only diagnostic messages and messages output using the .tm request are output.

-F*dir* Searches the directory *dir* for font width tables instead of using the system-dependent default directory.

-m*name* Prepends the macro file /usr/lib/tmac/tmac.*name* to the input *filename*s. Note that most references to macro packages include the leading m as part of the name; for example, the man macro package resides in /usr/lib/tmac/tmac.an.

-n*N* Numbers first generated page *N*.

-o*list* Prints only pages whose page numbers appear in the comma-separated *list* of numbers and ranges. A range *N*-*M* means pages *N* through *M*; an initial -*N* means from the beginning to page *N*; and a final *N*- means from *N* to the end.

-r*aN* Sets register *a* (1 character) to *N*.

-s*N* Stops the phototypesetter every *N* pages. On some devices, troff produces a trailer so you can change cassettes; resume by pressing the typesetter's start button.

-T*dest* Prepares output for typesetter *dest*. Only PostScript (-Tpsc) devices are currently supported.

`-u`*N* Sets the emboldening factor for the font mounted in position 3 to *N*. If *N* is missing, sets the emboldening factor to 0.

filename Specifies the file to be interpreted.

FILES

`/usr/lib/tmac/tmac.*`	Pointers to standard macro files
`/usr/lib/macros/*`	Standard macro files
`/usr/lib/font/devpsc`	Font width tables

SEE ALSO

`checknr`(1), `chmod`(1), `eqn`(1), `lpr`(1B), `nroff`(1), `tbl`(1)

`man`(7D), `me`(7D), `ms`(7D) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

`lpd`(8) in the *UNICOS Administrator Commands Reference Manual*, Cray Research publication SR–2022

NAME

`true`, `false` – Provides truth values

SYNOPSIS

```
true
false
```

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The `true` utility does nothing; it is always successful. It returns 0 as an exit code. `false` does nothing; it is always unsuccessful. It returns a nonzero as an exit code.

EXIT STATUS

The `true` utility has exit status 0; `false` has nonzero.

EXAMPLES

The following standard shell script executes the loop indefinitely:

```
while true
do
    command
done
```

SEE ALSO

`sh`(1)

NAME

tset, reset – Terminal-dependent initialization

SYNOPSIS

/usr/ucb/tset [-A] [-e*c*] [-E*c*] [-I] [-k*c*] [-n] [-Q] [-r] [-s] [-S] [-m [*ident*] [*test baudrate*]:*type*] [*type*] [-]

/usr/ucb/reset [-A] [-e*c*] [-E*c*] [-I] [-k*c*] [-n] [-Q] [-r] [-s] [-S] [-m [*ident*] [*test baudrate*]:*type*] [*type*] [-]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The tset utility may be used to set up your terminal when you first log in to a UNICOS system. It performs terminal-dependent processing such as setting erase and kill characters, setting or resetting delays, and sending any sequences needed to properly initialize the terminal.

It first determines the *type* of terminal involved, and then performs necessary initializations and mode settings. Type names for terminals can be found in the terminfo(5) database. If a port is not wired permanently to a specific terminal (not hardwired), it is given an appropriate generic identifier such as dialup.

The tset utility accepts the following options:

-A Asks user for default terminal type again. Used interactively to change terminal types without logging out and in again, or rereading the .profile file (sh(1)) or the .login file (csh(1)). For example:

```
$ TERM=xterm
     ...
$ tset -A
$ TERM = (xterm)
```

This allows a user to rerun tset as if a new login has been executed.

-e*c* Sets the erase character to the named character *c* on all terminals. The default is the backspace character on the terminal, usually <CONTROL-h>. The character *c* can either be typed directly, or entered using the hat notation used here.

-E*c* Sets the erase character to *c* on all terminals except those that cannot backspace (such as a TTY 33). *c* defaults to <CONTROL-h>.

-I Suppresses transmitting terminal initialization strings.

-k*c* Sets the line kill character to the named character `c` on all terminals. *c* defaults to `<CONTROL-u>`. The kill character is left alone if `-k` is not specified. The hat notation can also be used for this option.

`-n` On systems with the Berkeley 4BSD tty driver, `-n` specifies that the new tty driver modes should be initialized for this terminal. For a CRT, the `CRTERASE` and `CRTKILL` modes are set only if the baud rate is 1200 or greater. See `tty`(4) for more details.

`-Q` Quiet. Suppresses printing the `Erase set to` and `Kill set to` messages.

`-r` Reports terminal type.

`-s` Outputs `setenv` commands for `TERM`. This can be used with the following:

```
'tset -s ...'
```

This is preferred to using the following:

```
setenv TERM 'tset - ...'
```

`-S` Similar to `-s`, but outputs two strings suitable for use in `csh .login` files as follows:

```
set noglob
set term=('tset -S .....')
setenv TERM $term[1]
setenv TERMCAP "$term[2]"
unset term
unset noglob
```

`-m` [*ident*] [*test baudrate*]:*type*]
Specifies terminal type on ports that are not hardwired.

`-` Specifies the name of the terminal finally decided upon to be output on the standard output. This is intended to be captured by the shell and placed in the `TERM` environment variable.

If you do not specify any arguments, `tset` simply reads the terminal type out of the `TERM` environment variable and reinitializes the terminal.

When used in a startup procedure (`.profile` for `sh`(1) or `.login` for `csh`(1) users) it is desirable to provide information about the type of terminal you usually use on ports that are not hardwired. These ports are identified as `dialup`, `plugboard`, or `arpanet`, and so on.

To specify what terminal type you usually use on these ports, the `-m` (map) option is followed by the appropriate port type identifier (*ident*), an optional baud rate specification (*baudrate*), and the terminal type (*type*). (The effect is to "map" from some conditions to a terminal type, that is, to tell `tset` "If I am on this kind of port, assume that I am on that kind of terminal.") If you specify more than one mapping, the first applicable mapping prevails. A missing port type identifier matches all identifiers. Any of the alternate generic names given in `terminfo` can be used for the identifier.

A *baudrate* is specified as with stty(1), and is compared with the speed of the diagnostic output (which should be the control terminal). The baud rate *test* may be any combination of the >, @, <, and ! characters; the @ character means ''at'' and the ! character inverts the sense of the test. To avoid problems with metacharacters, it is best to place the entire argument to -m within \ characters; csh(1) users must also put a \ character before any ! character used here.

The following example maps baud rates and terminal types:

```
tset -m ´dialup>300:adm3a´ -m dialup:dw2 -m ´plugboard:?adm3a´
```

The preceding example, placed in the .profile file, causes the terminal type to be set to adm3a if the port in use is a dialup at a speed greater than 300 baud; to dw2 if the port is a dialup (that is, at 300 baud or less). If the *type* finally determined by tset begins with a question mark, you are asked if you really want that type. A null response means you want that type; otherwise, you can enter another type to be used. In this case, you will be queried on a plugboard port as to whether you are actually using an adm3a.

If no mapping applies and you specify a final *type* option (not preceded by a -m) on the command line, that type is used; otherwise, the identifier found in the environment is taken to be the terminal type. This should always be the case for hardwired ports.

It is usually desirable to return the terminal type, as finally determined by tset, and information about the terminal's capabilities to the shell environment. You can do this by using the - option.

When using the standard or Korn shell, put the following command in your .profile file:

```
eval ‘tset -s options...‘
```

When using the C shell, put the following command in your .login file:

```
eval ‘tset -s options...‘
```

Using the C shell, it is also convenient to create the following alias in your .cshrc file:

```
alias tset ’eval ‘tset -s \!*‘’
```

This alias allows the following command to be invoked at any time from your login csh.

```
tset 2621
```

Note to standard and Korn shell users: It is not possible to get this aliasing effect with a shell procedure because shell procedures cannot set the environment of their parent.

These commands cause tset to place the name of your terminal in the variable TERM in the environment.

When the terminal type is known, tset engages in terminal driver mode setting. This normally involves sending an initialization sequence to the terminal, setting the single-character erase (and optionally the line-kill (full-line erase)) characters, and setting special character delays. Tab and newline expansion are turned off during transmission of the terminal initialization sequence.

On terminals that can backspace but not overstrike (such as a CRT), and when the erase character is the default erase character (# on standard systems), the erase character is changed to backspace (<CONTROL-h>).

If `tset` is invoked as `reset`, it sets cooked and echo modes, turns off cbreak and raw modes, turns on newline translation, and restores special characters to a sensible state before any terminal-dependent processing is done. Any special character found to be NULL or –1 is reset to its default value.

This is most useful after a program terminates and leaves a terminal in an unusual state. You may have to type <LF>reset<LF> to get it to work because <CR> may not work in this state. Often none of this will echo.

EXAMPLES

The following examples are not interactive. They are lines to be included in the `.profile` file. These examples all assume the standard or Korn shell and use the `-` option. If you use `csh`(1), use one of the variations described above. A typical use of `tset` in a `.profile` or `.login` file will also use the `-e` and `-k` options, and often the `-n` or `-Q` options as well.

Example 1: Assume you are on a 2621. This is suitable for typing by hand but not for a `.profile`, unless you are always on a 2621.

```
export TERM; TERM=`tset - 2621`
```

Example 2: You have an h19 at home that you dial up on, but your office terminal is hardwired and known in `/etc/ttytype`.

```
export TERM; TERM=`tset - -m dialup:h19`
```

Example 3: You have a switch that connects everything to everything, making it nearly impossible to indicate on which port you are coming. You use a VT100 in your office at 9600 baud, and dial up to switch ports at 1200 baud from home on a 2621. Sometimes you use someone else's terminal at work, so you want it to ask you to make sure what terminal type you have at high speeds, but at 1200 baud you are always on a 2621. Note the placement of the question mark, and the quotes to protect the greater than and question mark from interpretation by the shell.

```
export TERM; TERM=`tset - -m 'switch>1200:?vt100' -m 'switch<=1200:2621''
```

Example 4: All of the above entries will fall back on the terminal type specified in the environment if none of the conditions hold. The following entry is appropriate if you always dial up at the same baud rate on different kinds of terminals. Your most common terminal is an adm3a. It always asks you what kind of terminal you are on, defaulting to adm3a.

```
export TERM; TERM=`tset - ?adm3a`
```

Example 5: If the environment is not properly initialized and you want to key entirely on the baud rate, the following can be used:

```
export TERM; TERM=`tset - -m '>1200:vt100' 2621`
```

Example 6: The following example illustrates the power of `tset`. You dial up at 1200 baud or less on a concept100, sometimes over switch ports and sometimes over regular dialups. You use various terminals at speeds higher than 1200 over switch ports, most often the terminal in your office, which is a VT100. However, sometimes you log in from the university over the ARPANET; in this case you are on an ALTO emulating a dm2500. You also often log in on various hardwired ports, such as the console, all of which are properly entered in the environment. You want your erase character set to <CONTROL-h>, your kill character set to <CONTROL-u>, and do not want `tset` to print the `Erase set to Backspace, Kill set to Control U`message. The following is to be typed as one continuous command line or enter the \ character to continue after pressing the <RETURN> key.

```
export TERM; TERM=`tset -e -k^U -Q - -m 'switch<=1200:concept100' -m\
'switch:?vt100' -m dialup:concept100 -m arpanet:dm2500`
```

FILES

`/usr/lib/terminfo`
Terminal capability database

SEE ALSO

`csh`(1), `sh`(1), `stty`(1)

`terminfo`(5) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

NAME

`tsort` – Performs a topological sort

SYNOPSIS

`tsort` [*file*]

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The `tsort` utility produces on the standard output a totally ordered list of items consistent with a partial ordering of items mentioned in the input *file*. If *file* is not specified, the standard input is used.

The input consists of pairs of items (nonempty strings) separated by blanks. Pairs of different items indicate ordering. Pairs of identical items indicate presence, but not ordering.

NOTES

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
`system`, `secadm`	Allowed to sort using any input file. In a privileged administrator shell environment, shell-redirected I/O is not subject to file protections.
`sysadm`	Allowed to sort using any input file subject to security label restrictions. Shell-redirected I/O is subject to security label restrictions.

Because `segldr`(1) and `ld`(1) perform multiple passes over relocatable object files, the use of `lorder`(1) and `tsort` is of little value.

If the `PRIV_SU` configuration option is enabled, the super user is allowed to sort using any input file. Shell-redirected I/O on behalf of the super user is not subject to file protections.

MESSAGES

`cycle in data`	A topological sort is not possible. The pairs causing the difficulty are printed to `stderr`.
`odd data`	An odd number of fields is in the input file.

BUGS

The `tsort` utility uses a quadratic algorithm, which is not worth fixing for the typical use of ordering a library archive file.

SEE ALSO

`ar`(1), `ld`(1), `lorder`(1), `segldr`(1)

NAME

`tty` – Prints the pathname of the user's terminal

SYNOPSIS

`tty [-l] [-s]`

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4
AT&T extensions (`-l` option)

DESCRIPTION

The `tty` utility prints the pathname of the user's terminal.

The `tty` utility accepts the following options:

`-l` Prints the synchronous line number to which the user's terminal is connected if it is on an active synchronous line.

`-s` Inhibits the printing of the terminal's pathname, allowing just the exit code to be tested.

NOTES

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
`system`, `secadm`	In a privileged administrator shell environment, allowed to write shell-redirected output to any file.
`sysadm`	Shell-redirected output is subject to security label restrictions.

If the `PRIV_SU` configuration option is enabled, the super user can write shell-redirected output to any file.

EXIT STATUS

The `tty` utility exits with one of the following values:

0 Standard input is a terminal.

1 Standard input is not a terminal.

>1 An error occurred.

MESSAGES

`not a tty`
: The standard input is not a terminal and `-s` is not specified.

`not on an active synchronous line`
: The standard input is not a synchronous terminal and `-l` is specified.

NAME

`type` – Writes a description of a command type

SYNOPSIS

`type` *name...*

IMPLEMENTATION

All Cray Research systems

STANDARDS

XPG4

DESCRIPTION

The `type` utility indicates how each argument would be interpreted if used as a command name.

The `type` utility accepts the following operands:

name Specifies a name to be interpreted.

The following environment variables affect the execution of `type`:

`LANG`	Provides a default value for the internationalization variables that are unset or null. If `LANG` is unset or null, the corresponding value from the implementation-specific default locale will be used. If any of the internationalization variables contains an invalid setting, the utility will behave as if none of the variables had been defined.
`LC_ALL`	If set to a nonempty string value, overrides the values of all the other internationalization variables.
`LC_CTYPE`	Determines the locale for the interpretation of sequences of bytes of text data as characters (for example, single- as opposed to multibyte characters in arguments).
`LC_MESSAGES`	Determines the locale that should be used to affect the format and contents of diagnostic messages written to standard error.
`NLSPATH`	Determines the location of message catalogs for the processing of `LC_MESSAGES`.
`PATH`	Determines the location of *name*, as described in the XBD specification, Chapter 6, Environment Variables.

The standard output of `type` contains information about each operand in an unspecified format. The information provided typically identifies the operand as a shell built-in, function, alias, or keyword, and where applicable, may display the operand's pathname.

Since `type` must be aware of the contents of the current shell execution environment (such as the lists of commands, functions, and built-ins processed by `hash`(1)), it is always provided as a shell regular built-in. If it is called in a separate utility execution environment, such as one of the following:

```
nohup type writer
find . -type f | xargs type
```

it might not produce accurate results.

EXIT STATUS

The `type` utility exits with one of the following values:

0 Successful completion.

>0 An error occurred.

SEE ALSO

`command`(1)

NAME

udbsee – Writes the ASCII source of a user database

SYNOPSIS

udbsee [-g] [-l] [-q] [-s] [-u] [-v] [-o *filename*] [-p *udb_path*] [*id*]

udbsee -a [-l] [-q] [-s] [-u] [-v] [-o *filename*] [-p *udb_path*]

udbsee -d [-l] [-q] [-s] [-u] [-v] [-o *filename*] [-p *udb_path*]

udbsee -f *fields* [-l] [-q] [-u] [-e '*field op expression*'] [-m *print_mask*] [-o *filename*] [-p *udb_path*] [*id*]

udbsee -f *fields* -a [-l] [-q] [-u] [-e '*field op expression*'] [-m *print_mask*] [-o *filename*] [-p *udb_path*]

udbsee -f *fields* -d [-l] [-q] [-u] [-e '*field op expression*'] [-m *print_mask*] [-o *filename*] [-p *udb_path*]

udbsee -G [-l] [-o *filename*] [-p *udb_path*]

udbsee -h

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The udbsee utility converts information from the user database (UDB) into an ASCII file. This file presents to administrators user information in a readable form, provides a vehicle by which minor changes to user information can be made with an ordinary editor and major changes may be made with the existing UNICOS text processing filters, allows easy reconstruction of the UDB, and allows users to easily view their own control parameters. The current values for the global tape name map and the UDB defaults will be included in the output only if the -a, -d, or -g option is used.

The output of the first, second, and third forms of udbsee (see the SYNOPSIS section) is UDB source accepted by udbgen(8). The output of the fourth, fifth, and sixth forms (using *print_mask*) can be specified by users as described below.

The last form of udbsee is informational and writes the usage messages and a list of the UDB fields to the stderr file.

The general format of the udbsee display is field_name :field_value:. Generally the field names and values should be meaningful, but see udbgen(8) for definitions of names and possible values. If the # character appears anywhere except within a field value, all characters from the # to the end of line will be ignored by udbgen.

Only appropriately authorized users are shown sensitive information from the protected user database.

Options can appear in any order as long as they all appear before *filename* or *id*. Restrictions are shown in the SYNOPSIS section for the various forms of the utility.

The `udbsee` utility accepts the following options:

`-a` Displays all records in the database in ascending user identification (UID) order. This option is not allowed when `-d`, *filename*, or *id* is present. By default, only named records are displayed.

`-d` Displays all records in the database in disk order. This option is not allowed when `-a` or *id* is present. By default only named records are displayed. This option should be used instead of `-a` if the database is corrupted and you want to get as much of the source as possible for regeneration.

`-e '`*field op expression*`'`
For each entry in the UDB selected with `-a`, `-d`, or an *id* list, evaluates the *expression* argument. If the result is nonzero, the record will be selected to be printed according to the `-m` option. The *field* arguments are any of the field names listed with the `-f` option.

The *op* arguments are as follows:

`||` Binary OR. Nonzero if the left-hand side or the right-hand side evaluates to a nonzero value.

&& Binary AND. Nonzero if the left-hand side and the right-hand side are both nonzero.

== != Equal/not equal to. Nonzero if the left-hand side and the right-hand side are equal/not equal.

> < Greater/less than. Nonzero if the left-hand side is greater/less than the right-hand side.

>= <= Greater/less than or equal to. Nonzero if the left-hand side is greater/less than or equal to the right-hand side. For example, `uid >= 100 uid <= 200` would select all records with UID values in the range 100 through 200.

~ Regular expression matching. Nonzero if the string on the left-hand side matches the regular expression given by the string on the right-hand side. Regular expressions are given in the style of `ed`(1). For example, `name ~ "^[a-c].*"` would select all records with user names beginning with `a`, `b`, or `c`.

! Unary not. Nonzero if the right-hand side evaluates to zero.

".." A string of characters.

{..} A date. Date specifications are in the style `{[[[[[yy]mm]dd]hh]mm][.ss]}`. For example, `{01271200}` would be noon on the 27th of January in the current year.

(..) A subexpression.

Note that the expression may have to be quoted to stop the shell from interpreting symbols (for example, &) as symbols having special meaning.

`-f` *fields*
Field list. One or more UDB field names must be provided as comma-separated strings. If the `-m` option is not also specified, the first field name must be `create`, `delete`, or `update`. That string is first on the output line and is followed by the user name (login) within colons. Each remaining named field is output in the listed order, using the format *'name :value:'*. Each field name is separated from its predecessor by a single blank. A new line occurs at the end of the list. The special word `all` may be the terminal name in the list and causes all fields not yet written to appear in an internally specified order in the output file. (See the `-h` option for a way to view this order.) When the `-m` option is specified, no specific requirements other than the field names being recognized are imposed. In this case, the special word `all` is not recognized and field names, colons, and new-line characters are not automatically provided. Strings `create`, `delete`, `name`, and `update` always refer to the field known as `name` in the database.

`-g` Prints global tables. This option is not allowed with the `-a`, `-d`, or `-f` option. Global tables are always printed when `-a` or `-d` is used without the `-f` option.

`-G` Prints only the global tables. This option is used to print the global defaults independent of any user records. The only options allowed with the `-G` option are `-l`, `-p`, and `-o`. In addition, no *id*s can be specified on the command line.

`-h` Causes all other options to be ignored and writes to the `stderr` file a usage message showing the options and a table of the internal field names and their default conversion specifications. The order of the names in the table is the natural order in which the fields are written when the `all` field name is used. This is the same as a usage error, except that the program exits with a value of 0.

`-l` Locks the database while reading. This option is useful when a full source file is needed, but updates by other users could be going on at the same time.

`-m` *print_mask*
Prints mask (requires the `-f` option). This provides a way to control the format of the output. The print mask is presented as a format to `fprintf`(3C) when the output is written.

`-o` *filename*
Writes output to *filename*. When this option is not present, `stdout` is used.

`-p` *udb_path*
Sets the path name for access to the UDB files; the default path name is /`etc`. This option allows private or test versions of the UDB files to be created and maintained. The path name must end with a directory name; if it does not, it will be declared incorrect. For example, to specify that the UDB files are to be in `/c/abc/udbtest`, enter `-p /c/abc/udbtest` on the command line. When `-p` is specified, read access privilege to the protected database file is required.

`-q` Quiet. Suppresses informative messages and eliminates the comment lines in the normal output format.

`-s` Suppresses sensitive fields in the output. The administrator can use this to create a public version of the source. An ordinary user does not have access to sensitive fields in environments that are established as suggested. When the `-f` option is specified, this option is illegal.

`-u` Displays database access statistics.

`-v` Shows all UDB fields. If this option is not present, only fields that contain non-zero values will be shown in the output. This option presents a predictable sequence of output to simplify automated processing. On a secure system, for users who do not have security administrator privileges, security fields are printed showing zeros (meaning having no privileges), regardless of their actual authorized attributes. This is due to accessing of the public version of the UDB, which does not contain security information. When the `-f` option is specified, this option is not valid.

id Displays the UDB record belonging to *id*. *id* can be either a user name or a user ID (UID). This argument is illegal with the `-a` or `-d` option. If no record is found for the user name and the name is numeric, an attempt will be made to find a record with a UID equal to that number. Mixed user names and UIDs are allowed in *id* lists because `udbsee` evaluates the type of each ID as it is encountered in the option list. Any number of names or UIDs may be present, up to the limit of the command-line length; each name or UID is taken in order. If this option is not present, the record belonging to the login of the running process will be accessed.

NOTES

If this utility is installed with a privilege assignment list (PAL), a user who is assigned the following privilege text upon execution of this command is allowed to perform the action shown:

Privilege Text	Action
`seeany`	Allowed to retrieve sensitive information from the protected user database.

If this utility is installed with a PAL, a user with one of the following active categories is allowed to perform the action shown:

Active Category	Action
`system`, `secadm`, `sysadm`	Allowed to retrieve sensitive information from the protected user database.

If the `PRIV_SU` configuration option is enabled, the super user is allowed to retrieve sensitive information from the protected user database.

BUGS

If the file produced by use of the `-s` option or by a user not privileged to access the private UDB files (see the FILES section) is used to create a UDB, all sensitive fields will be set to their default values.

EXAMPLES

In the following example, assume that user `fil` is defined in the user database (user input is shown in bold):

```
$ udbsee fil
create  :fil:   uid     :64:
        comment :F. I. Ling:
        passwd  :TI..ekLKRkiZI:
        pwage   :-force, -super user,
                 4w, 1d, 703461565: #Max, min age, changed 04/16/92 16:59:25
        gids    :102,   64,     1,      2,      3,
                 14,    15,     16:
        acids   :61,    16,     14,     13,     12,
                 1:
        dir     :/w/fil:
        resgrp          :0: # uid
        jproclim[b]     :98:
        jcpulim[b]      :unlimited:
        jmemlim[b]      :unlimited:
        jfilelim[b]     :unlimited:
        pcpulim[b]      :unlimited:
        pmemlim[b]      :unlimited:
        pfilelim[b]     :unlimited:
        jproclim[i]     :98:
        jcpulim[i]      :unlimited:
        jmemlim[i]      :unlimited:
        jfilelim[i]     :unlimited:
        pcpulim[i]      :unlimited:
        pmemlim[i]      :unlimited:
        pfilelim[i]     :unlimited:
```

For definitions of the field names, and possible values for each, see udbgen(8).

To write the ASCII source of the entire database in its long form to a file named dbsource, enter the following command:

```
$ udbsee -av -o dbsource
```

The following command displays output for the UDB entry for user fil, shown previously:

```
$ udbsee -f update,dir,uid fil
update :fil: dir :/w/fil: uid :64:
```

The -e option lets you match an element in a list with an exact string. For example, if you want to select all users with a group ID (GID) of 4, specify the following:

```
$ udbsee -a -f update,gids -e'gids~"^([0-9]+,)*4(,[0-9]+)*$"'
```

FILES

`/etc/udb`	User validation file containing user control limits
`/etc/udb.public`	Public version of the user database file
`/etc/udb_2/udb.index`	Public extension file index
`/etc/udb_2/udb.priva`	Private field extension file
`/etc/udb_2/udb.pubva`	Public field extension file

SEE ALSO

`ed`(1), `privtext`(1)

`printf`(3C) in the *UNICOS System Libraries Reference Manual*, Cray Research publication SR–2080

`udbgen`(8) in the *UNICOS Administrator Commands Reference Manual*, Cray Research publication SR–2022

General UNICOS System Administration, Cray Research publication SG–2301

NAME

`ulimit` – Sets or reports file size limit

SYNOPSIS

`ulimit` [`-f`] [*blocks*]

IMPLEMENTATION

All Cray Research systems

STANDARDS

XPG4

DESCRIPTION

The `ulimit` utility sets or reports the file-size writing limit imposed on files written by the shell and its child processes (files of any size may be read). Only a process with appropriate privileges can increase the limit.

The `ulimit` utility accepts the following options and operands:

`-f` Sets (or reports, if no *blocks* operand is present) the file size limit in blocks. The `-f` option is also the default case.

blocks The number of 512-blocks to use as the new file size limit.

The standard output is used when no *blocks* operand is present. If the current number of blocks is limited, the number of blocks in the current limit is written in the following format:

`"%d\n", <`*number of 512-byte blocks*`>`

If there is no current limit on the number of blocks, in the POSIX locale the following format is used:

```
"unlimited\n"
```

Since `ulimit` affects the current shell execution environment, it is always provided as a shell regular built-in. If it is called in separate utility execution environment, such as one of the following:

```
nohup ulimit -f 10000
env ulimit 10000
```

it will not affect the file size limit of the caller’s environment.

Once a limit has been decreased by a process, it cannot be increased (unless appropriate privileges are involved), even back to the original system limit.

The following environment variables affect the execution of `ulimit`:

`LANG`	Provides a default value for the internationalization variables that are unset or null. If `LANG` is unset or null, the corresponding value from the implementation-specific default locale will be used. If any of the internationalization variables contains an invalid setting, the utility will behave as if none of the variables had been defined.
`LC_ALL`	If set to a non-empty string value, override the values of all the other internationalization variables.
`LC_CTYPE`	Determines the locale for the interpretation of sequences of bytes of text data as characters (for example, single- as opposed to multibyte characters in arguments).
`LC_MESSAGES`	Determines the locale that should be used to affect the format and contents of diagnostic messages written to standard error.
`NLSPATH`	Determines the location of message catalogues for the processing of `LC_MESSAGES`.

EXIT STATUS

The following exit values are returned:

0 Successful completion.

>0 An error occurred.

SEE ALSO

`sh`(1)

NAME

umask – Sets file-creation mode mask

SYNOPSIS

umask [-S] [*mask*]

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The user file-creation mode mask of the current shell execution environment is set to the value specified by the *mask* operand.

If you omit *mask*, the current value of the mask is printed.

The umask utility accepts the following option and operand:

-S Product symbolic output, which is in the form:

u=<*owner permissions*>, g=<*group permissions*>, o=<*other permissions*>

mask A string that specifies the new file mode creation mask. The string is treated in the same way as the *mode* operand described in chmod(2); *mask* may be represented as three octal digits or as a symbolic mode.

The octal digits refer to read, write, and execute permissions for *owner*, *group*, and *others*, respectively (see chmod(2) and umask(2)). The value of each specified octal digit is first logically complemented and then logically AND'ed with the corresponding "digit" specified by the system for the creation of a file (see creat(2)).

For a symbolic mode value, the new value of the file mode creation mask is the logical complement of the file permission bits portion of the file mode specified by the symbolic mode string.

In a symbolic mode value, the permissions *op* characters + and - are interpreted relative to the current file mode creation mask. A + character clears the bits for the indicated permissions in the mask. A - character sets the bits for the indicated permissions in the mask.

As in chmod, application use of the octal number form for the *mode* values is obsolescent.

You can include umask in your .profile file (see profile(5)) and it can be invoked at login to set automatically your permissions on files or directories created. See the NOTES section.

NOTES

The umask utility is a built-in utility to the standard shell sh(1). An executable version of this utility is available in /usr/bin/umask.

The default setting for umask is 077. You will be affected by this setting in the following ways if you do not have a umask specified in your $HOME/.profile or $HOME/.login files:

- If you assume that new files automatically have group and/or world access, you must manually set the file access permission bits to ensure the appropriate type of access is available.
- If you want a umask other than 077, you must place the umask command in your $HOME/.profile or $HOME/.login file.

EXIT STATUS

The umask utility exits with one of the following values:

0 The file mode creation mask was successfully changed, or no *mask* operand was supplied.

>0 An error occurred.

EXAMPLES

Example 1: The following command removes write permission for *group* and *others* (files usually created with mode 777 become mode 755, and files created with mode 666 become mode 644):

```
umask 022
```

SEE ALSO

chmod(1), sh(1),

chmod(2), creat(2), umask(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

profile(5) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

NAME

unalias – Removes alias definitions

SYNOPSIS

```
unalias alias-name ...
unalias -a
```

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The unalias utility removes the definition for each alias name specified (see sh(1)). The aliases are removed from the current shell execution environment.

The unalias utility accepts the following option and operand:

alias-name Denotes the name of an alias to be removed.

-a Remove all alias definitions from the current shell execution environment.

NOTES

The unalias utility is a built-in utility to the standard shell (sh(1)). An executable version of this utility is available in /usr/bin/unalias.

The csh(1) utility has a built-in unalias utility with slightly different characteristics. See csh(1).

EXIT STATUS

The unalias utility exits with one of the following values:

0 Successful completion.

>0 One of the *alias-name* operands specified did not represent a valid alias definition, or an error occurred.

SEE ALSO

alias(1), csh(1), sh(1)

NAME

uname – Prints name of current system

SYNOPSIS

uname [-a] [-m] [-n] [-r] [-s] [-v]

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The uname utility prints the current system name on the standard output file. It is useful mainly to determine the name of the system that you are using.

The uname utility accepts the following options, which print selected information returned by uname.

-a Prints information in the following format:

system node release version hardware

-m Prints the machine's hardware name.

-n Prints the node name (the node name may be a name by which a system is known to a communications network).

-r Prints the operating system release number.

-s Prints the system name (default).

-v Prints the operating system version number.

NOTES

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
system, secadm	In an administrator privileged shell environment, shell-redirected I/O is not subject to security label restrictions.

If the PRIV_SU configuration option is enabled, and if the user is the super user, shell-redirected I/O is not subject to security label restrictions.

EXIT STATUS

The uname utility exits with one of the following values:

0 The requested information was written successfully.

>0 An error occurred.

EXAMPLES

The following example prints all uname information about the system:

```
$ uname -a
snq1 snq1 UNICOS aj CRAY-YMP
```

SEE ALSO

uname(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

UNICOS Configuration Administrator's Guide, Cray Research publication SG–2303

NAME

uncompress – Expands expanded files

SYNOPSIS

```
uncompress [-c] [-f] [-v] [filename]
uncompress [-f] [-v] [filename ...]
uncompress -P
```

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4
AT&T

DESCRIPTION

The uncompress utility will restore files to their original state after they have been compressed using the compress(1) utility. If no files are specified, the standard input wil be uncompressed to the standard output. If the invoking process has appropriate privileges, the ownership, modes, access time, and modification time of the original file are preserved.

This utility supports the uncompressing of any files produced by the compress(1) utility on the same implementation. For files produced by compress(1) on any other systems, uncompress supports 9- to 14-bit compression (see the -b option of compress(1)); it is implementation-dependent whether values of -b greater than 14 are supported.

The uncompress utility accepts the following options:

-c Writes to the standard output; no files are changed. The nondestructive behavior of zcat(1) is identical to that of specifying uncompress -c.

-f Does not prompt for overwriting files. Except when run in the background, if -f is not given, the user will be prompted as to whether an existing file should be overwritten. If the standard input is not a terminal and -f is not given, uncompress will write a diagnostic message to standard error and exit with a status greater than zero.

-v Verbose. Displays the percentage reduction for each file compressed.

-P (Only affects uncompress of input from standard input.) If -P is not specified and the input is not compressed, a message is output and the command terminates with an exit status of 1. With -P, the message is suppressed, the input is copied to the standard output, and the command terminates with an exit status of 0.

NOTES

Although compressed files are compatible between machines with large memory, `-b 12` should be used for file transfer to architectures with a small process data space (64 Kbytes or less).

The `compress` utility should be more flexible about the existence of the `.Z` suffix.

EXIT STATUS

The following exit values are returned:

0 Successful completion.

>0 An error occurred.

MESSAGES

filename`: already exists; do you wish to overwrite (y or n)?`
Respond `y` if you want the output file to be replaced; `n` if not.

`uncompress: corrupt input`
A `SIGSEGV` violation was detected, which usually means that the input file is corrupted.

SEE ALSO

`compress`(1), `pack`(1), `sh`(1), `zcat`(1)

''A Technique for High Performance Data Compression,'' Terry A. Welch, *IEEE Computer*, vol. 17, no. 6 (June 1984), pp. 8–19.

NAME

unget – Undoes a previous get of an SCCS file

SYNOPSIS

unget [-n] [-r *SID*] [-s] *files*

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The unget utility, used before creating the intended new delta, undoes the effect of a get -e. If a directory is named, unget behaves as though each file in the directory were specified as a named file, except that non-SCCS files and unreadable files are silently ignored. If a name of - is specified, the standard input is read with each line being taken as the name of a Source Code Control System (SCCS) file to be processed.

Options apply independently to each named file. The valid options are as follows:

-n Causes the retention of the selected file, which would normally be removed from the current directory.

-r *SID* Uniquely identifies which delta is no longer intended. (This would have been specified by get(1) as the "new delta"). The use of this option is necessary only if two or more outstanding get(1) commands for editing on the same SCCS file were done by the same person (login name). A diagnostic results if the specified *SID* is ambiguous, or if it is necessary and omitted on the command line.

-s Suppresses the printout, on the standard output, of the intended delta's *SID*.

files Specifies the file to be restored.

EXIT STATUS

The unget utility exits with one of the following values:

0 Successful completion.

>0 An error occurred.

MESSAGES

Error messages from SCCS are printed. Use help(1) for explanations.

SEE ALSO

admin(1), cdc(1), comb(1), delta(1), get(1), help(1), prs(1), rmdel(1), sact(1), sccsdiff(1), val(1), vc(1), what(1)

sccsfile(5) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

NAME

unifdef – Resolves and removes lines surrounded by #ifdef preprocessor statements

SYNOPSIS

/usr/ucb/unifdef [-c] [-l] [-t] [-D*name*] [-U*name*] [-iD*name*] [-iU*name*] [*file*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The unifdef utility removes lines surrounded by #ifdef preprocessor statements from *file*; it does not affect *file* in any other way. It deals appropriately with nested #ifdef statements, comments, and single and double quotation marks of C syntax, but it does not include or interpret macros. In addition, it does not strip out comments, although it recognizes and ignores them. You can specify which symbols you want defined by using the -D options, and which you want undefined by using the -U options. Lines within #ifdef preprocessor statements are either copied to the output or removed, as appropriate. Any #ifdef, ifndef, else, and endif statements associated with *name* are also removed.

#ifdef preprocessor statements involving symbols you did not specify remain and are copied along with their associated #ifdef, else, and endif statements.

When an #ifdef X statement is nested inside another #ifdef X statement, the inside #ifdef statement is treated as if it were an unrecognized symbol. If the same symbol appears in more than one argument, only the first occurrence will be significant.

The unifdef utility copies its output to standard output and takes its input from standard input if a file argument is not specified.

The unifdef utility accepts the following options:

-c Complements the normal operation. Lines that would have been removed or blanked are retained, and vice versa.

-l Replaces removed lines with blank lines.

-t Specifies plain text option. unifdef does not recognize comments, or single or double quotation marks.

Of the following options, one or more of each may be used, but at least one must be specified:

-D*name* Defines symbol *name*. (see previous discussion).

-U*name* Undefines symbol *name*. (see previous discussion).

-iD*name* Ignores but prints lines associated with defined symbol *name*. If you use #ifdefs to delimit non-C lines, such as comments or code under construction, you must specify which symbols are used for that purpose so that unifdef does not try to parse for quotation marks and comments within them.

`-iU`*name* Ignores but prints lines associated with undefined symbol *name*.

file Specifies the name of file to be processed.

CAUTIONS

Premature EOF signifies an inappropriate `else` or `endif` statement.

EXIT STATUS

The `unifdef` utility exits with one of the following values:

0 Output is exact copy of input.

1 Output is different from input.

2 Trouble.

BUGS

The `unifdef` utility does not know how to handle `cpp`(1) constructs such as the following:

```
#if defined(X) || defined(Y)
```

EXAMPLES

The following shows the contents of `file`:

```
hello() {
#ifdef _CRAY2
        printf("Hello cray20);
#else
        printf("Hello cray10);
#endif
}
```

The following command removes the `#ifdef` preprocessor statements from `file`.

```
$ unifdef -D_CRAY2 file
```

The following shows the contents of `file` after using the `unifdef` command.

```
hello() {
       printf("Hello cray20);
}
```

SEE ALSO

`cpp`(1), `diff`(1)

NAME

uniq – Reports repeated lines in a file

SYNOPSIS

uniq [-c] [-f *fields*] [-s *chars*] [*input_file* [*output_file*]]
uniq -d [-f *fields*] [-s *chars*] [*input_file* [*output_file*]]
uniq -u [-f *fields*] [-s *chars*] [*input_file* [*output_file*]]

Obsolescent version; may not be supported in future releases:

uniq [-c] [-*n*] [+*m*] [*input_file* [*output_file*]]
uniq -d [-*n*] [+*m*] [*input_file* [*output_file*]]
uniq -u [-*n*] [+*m*] [*input_file* [*output_file*]]

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The uniq utility reads the input and compares adjacent lines. By default, uniq removes the second and succeeding copies of repeated lines, and the remainder is written on the output file. When you omit *input_file*, or specify *input_file* as -, the standard input is used. When you omit *output_file*, output is written to standard output. To be found, repeated lines must be adjacent (see sort(1)).

The typical mode output is the union of the -u and -d mode output.

The uniq utility accepts the following options and operands:

-c Precedes each line with the number of times it occurred.

-f *fields* Ignores the first *fields* on each input line when doing comparisons; *fields* is a decimal integer. A field is the maximal string matched by the basic regular expression:

```
[[:blank:]]*[^[:blank:]]*
```

If the *fields* argument specifies more fields than appear on an input line, a null string is used for comparison.

-s *chars* Ignores the first *chars* characters when doing comparisons. *chars* is a decimal integer. Fields are skipped before characters. If you specify in conjunction with option the -f option, the first *chars* characters after the first *fields* are ignored. If the *chars* argument specifies more characters than remain on an input line, a null string is used for comparison.

-d Suppresses the writing of lines that are not repeated in the input.

`-u`	Suppresses the writing of lines that are repeated in the input.
`-`*n*	(Obsolescent) Equivalent to `-f` *fields* with *fields* set to *n*.
`+`*m*	(Obsolescent) Equivalent to `-s` *chars* with *chars* set to *m*.
input_file	File about which to be reported.
output_file	File that contains the results.

NOTES

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
`system`, `secadm`	In a privileged administrator shell environment, shell-redirected I/O is not subject to file protections.
`sysadm`	Shell-redirected output is subject to security label restrictions.

If the `PRIV_SU` configuration option is enabled, shell-redirected I/O on behalf of the super user is not subject to file protections.

The *input_file* and *output_file* operands should always be different files.

EXIT STATUS

The `uniq` utility exits with one of the following values:

0 Successful completion.

>0 An error occurred.

SEE ALSO

`comm`(1), `sort`(1)

NAME

units – Unit conversion program

SYNOPSIS

units

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The units utility converts quantities expressed in various standard scales to their equivalents in other scales. It works interactively in this fashion:

```
$ units
  You have:  inch
  You want:  cm
          *2.540000e+00
          /3.937008e-01
```

A quantity is specified as a multiplicative combination of units optionally preceded by a numeric multiplier. Powers are indicated by suffixed positive integers, division by the usual sign:

```
$ units
  You have:  15 lbs force/in2
  You want:  atm
          *1.020689e+00
          /9.797299e-01
```

The units utility does only multiplicative scale changes (for example, it can convert Kelvin to Rankine, but not Celsius to Fahrenheit). Most familiar units, abbreviations, and metric prefixes are recognized, along with the following:

pi	Ratio of circumference to diameter
c	Speed of light (meters/second)
e	Charge on an electron (coulombs)
g	Acceleration of gravity (meters/second^2)
force	Same as g
mole	Avogadro's number
water	Pressure head per unit height of water (meters/kilograms/second^2)
au	Astronomical unit (meters)

`lb` is recognized as a unit of mass; `pound` is not. Compound names are run together (for example, `lightyear`). British units that differ from their U.S. counterparts are shown with `br` as a prefix (for example, `brgallon`). For a complete list of units, enter the following:

```
cat /usr/lib/unittab
```

Press `<CONTROL-d>` to exit the program.

MESSAGES

`Conformability`	You specified mutually incompatible scales.
`Cannot recognize` *string*	Unknown keyword.

FILES

`/usr/lib/unittab`	File containing conversion table

NAME

ustat – Displays Unified Resource Manager (URM) session information

SYNOPSIS

ustat [-a] [-c *rate*] [-H *host*] [-h] [-l] [-m] [-O *category*] [-P] [-r] [-S *state*] [-s *service*] [-T *type*] [-v] [-W *reason*] [-w] [*login* [*login* ...]]

ustat [-h]

IMPLEMENTATION

Cray PVP systems

DESCRIPTION

The ustat utility displays session (job) information from the Unified Resource Manager (URM), including jobs queued by the Network Queuing System (NQS).

By default, you can display information for your jobs only; only the super user can see all jobs. However, the rmgr(1) command can be used to set up a group of users as *authorized users*; these users are allowed to see each others' job information. To allow all users to see all jobs on the system, the system administrator can set the rmgr(1) configuration object /ustat/showall to 1. For more information on setting up authorized users or changing the /ustat/showall configuration object, see the rmgr(1) man page.

When invoked with no options, ustat shows a short display of job information.

The ustat utility accepts the following options and operands:

-a Displays all users. Only authorized users and the super user can see other users' jobs; see the rmgr(1) man page for more information.

-c *rate* Runs a continuous display with the specified refresh rate (in seconds). Specify *rate* as 10 or greater; any value between 1 and 9 defaults to 10 seconds.

-H *host* Specifies the host machine (local by default).

-h Displays a usage statement and online help. It contains the site-specific names defined with the user exits, if any.

-l Specifies long format for the Why field; that is, expands the brief description of the reason that the job is in the current state.

-m Displays minimum rank of a job (called *interqueue priority* in NQS). Nonbatch jobs always display the default value of 0.

-O *category* Sorts output by one of the following categories:

Category	Description
name	Sorts output by user name; each user's jobs are also sorted by time of initiation.
rank	Sorts output by job priority (URM rank).

`resource` Sorts output by resources used. The abbreviation `res` can also be used.

`type` Sorts output by session initiator type. See `-T` option for a list of valid session initiators.

`-P` Displays the priority (URM rank) of jobs.

`-r` Displays requested resources for tape, secondary data segments (SDS), and massively parallel processing (MPP), along with the requested amount, in the rightmost column of output. An asterisk next to the resource name indicates that the request is unsatisfied.

`-S` *state* Limits display to sessions (jobs) in the specified state. Valid session states include:

State	Notes
`incomplete`	
`invalid`	
`none`	Job state is displayed as `no status`
`pend_delete`	Job state is displayed as `pending delete`
`ranked`	
`registered`	
`rejected`	
`restarted`	
`running`	
`selected`	
`svchold`	Job state is displayed as `service hold`
`urm_hold`	

`-s` *service* Specifies a URM socket (`urm` by default). This option is useful only for internal testing purposes; it is not needed in normal use.

`-T` *type* Limits display to sessions (jobs) matching the specified session initiator type. The following session initiators are valid: `batch`, `ftp`, `login`, `Unknown`, `rsh`, `rexec`, `cron`, `site1`, `site2`, and `site3`.

Note: Session initiators `site1`, `site2`, and `site3` are reserved for use with the user exit capability.

`-v` Displays information on current system memory. This information appears before the list of sessions (jobs) in the following format:

```
Memory Target (Megawords):   486.36
Memory  Load  (Megawords):   534.63
```

`-W` *reason* Limits display to sessions (jobs) matching the specified reason (the brief reason displayed in the `Why` field). To display the reason in a long format, use the `-l` option. Valid reasons include the following:

Reason	Description
`availres`	The requested resource is unavailable.

`bb_max`	The job exceeds the system-wide MPP barrier resource target.
`bb_tar`	The job exceeds the per-user MPP barrier resource limit.
`chkpnt`	A checkpoint operation has been recommended.
`cpu_max`	The job exceeds the CPU resource limit.
`fifo`	The job has received an intermediate rank (fifo); this is an internal state that is rarely displayed by `ustat`.
`invalid`	The reason code is invalid; this is an internal state that is rarely displayed by `ustat`.
`job_max`	The job exceeds the maximum number of jobs allowed on the system.
`job_tar`	The target job count has been reached.
`jstate`	The job is being held while the service provider is disabled; this reason is valid for the batch resource (and site-specific resources, if enabled).
`mem_max`	The job exceeds the maximum amount of memory for the system.
`mem_tar`	The target memory usage value has been reached.
`merit`	The job has been selected for recommendation.
`none`	No reason is available for this job. Specify `-W none` to display all jobs with no associated reason code.
`pe_max`	The job exceeds the system-wide MPP processing element (PE) resource maximum limit.
`pe_tar`	The target count for the MPP PE resource has been reached.
`petime_max`	The job exceeds the MPP PE time maximum.
`preempt`	Preemption has been recommended for this job; this status results in a checkpoint or suspension operation.
`poolres`	The job cannot be started in any MPP pool.
`recstart`	A start operation has been recommended.
`restore`	A restore operation has been recommended.
`rstart`	A restart operation has been recommended.
`sds_max`	The job exceeds the SDS maximum.
`sds_tar`	The target SDS usage has been reached.

`site1_tar`
`site2_tar`
`site3_tar`
	These reasons specify site-specific status, if available. Sites can change these reasons using the user exit capability.

`start_max`	The start limit (the number of jobs initiated during one URM cycle) has been reached for this service provider.
`svc_max`	A resource request exceeds the service limit.
`tape_max`	The job's tape request exceeds the maximum amount.
`tape_tar`	The target tape usage has been reached.
`when`	A *when condition* has not occurred (deferred implementation).

`-w` Specifies wide display; shows the memory and CPU resources that have been requested and used.

login Displays information for specified user (valid only for super user unless all users have been authorized)

EXAMPLES

Example 1: You can display information about your own jobs in short format by running `ustat` with no options. Note that if `rmgr`(1) is configured to allow all users to see all jobs, this utility displays all jobs on the system. (In this case, you can use the command `ustat` *my_login* to limit the display to your own jobs.)

```
$ ustat

User      Sid   Type    Age(mn) State       Why
----      ---   ----    ------- -----       ---
crs       345   login   7.8     running
```

The fields of `ustat` output have the following meaning:

Field	Description
`User`	Displays the user ID
`Sid`	Displays the session ID assigned by the system
`Type`	Displays the type of session initiator that started the job (see the `-T` option)
`Age`	Displays the time, in minutes, that the job has been registered with URM
`State`	Displays the current state of the job (see the `-S` option)
`Why`	Displays the reason for the state of the job (see the `-W` option)

Example 2: Running `ustat` with the `-a` option displays all information that you are authorized to see. By default, users that are not the super user see only their own jobs. However, if URM is configured to allow users to see all jobs, users will see all jobs on the system, as in the following example:

```
$ ustat -a

User      Sid   Type    Age(mn) State       Why
----      ---   ----    ------- -----       ---
rkb       15    login   28.0    running
darason   166   login   16.1    running     preempt
augs      169   login   15.9    running
darason   172   login   15.7    running
n3875     106   NQS     171.3   running
tzioufas  0     NQS     7.1     urm hold    mem_tar
root      0     NQS     4.0     urm hold    mem_tar
root      0     NQS     1.2     urm hold    mem_tar
```

`ustat` displays a reason for job status in the `Why` field, if a reason is available. This includes jobs that are held before being allowed to execute, waiting for the total system memory load to drop. In example 2, the reason for the job with session ID 166 is `preempt`, which signifies that the job is being preempted (suspended).

Example 3: Authorized users can specify the following command to display verbose information for all batch jobs on the system:

```
$ ustat -a -l -T batch

User      Sid   Type    Age(mn) State       Why
----      ---   ----    ------- -----       ---
n3875     106   NQS     171.6   running
elpoole   111   NQS     59.3    running
n4074     107   NQS     1743.2  running
n3875     113   NQS     454.3   running
n3825     0     NQS     159.2   urm hold    target memory usage reached
tzioufas  0     NQS     7.5     urm hold    target memory usage reached
root      0     NQS     4.3     urm hold    target memory usage reached
root      0     NQS     1.5     urm hold    target memory usage reached
```

Example 4: Authorized users can use the following command to display information for all batch jobs (`-a` and `-T`) on the system, sorted by resource use (`-O`), with an expanded reason in the `Why` field (`-l`), including columns for memory and CPU resources (`-w`), and preceded by a summary of current system memory (`-v`).

```
$ ustat -alwv -T batch -O res

Looking on hot using service port urm at users:
        all users...

          Memory Target (Megawords):      486.36
          Memory  Load  (Megawords):      534.63

User     Sid Type Age(mn) State    ReqMem UseMem ReqCpu UseCpu Why
----     --- ---- ------- -----    ------ ------ ------ ------ ---
n4074    107 NQS  1743.5  running  215.00 211.85    360      0
n3875    113 NQS  454.6   running  128.00 121.05    240      0
n3875    106 NQS  171.9   running   64.00  42.90     60     20
orapat   105 NQS  22.1    running    0.00  40.01      0     20
orapat   109 NQS  240.1   running  128.00  39.98     60     20
elpoole  111 NQS  59.6    running   22.20  20.39     60     19
slow     108 NQS  75.3    running   20.03   6.28    500     20
n2811    110 NQS  182.5   running    8.00   2.19   1440     20
hertr    0   NQS  17.2    urm hold  16.00   0.00    360      0 target memory usage reached
tzioufas 0   NQS  7.8     urm hold  20.20   0.00    240      0 target memory usage reached
n3825    0   NQS  159.6   urm hold 100.00   0.00    403      0 target memory usage reached
root     0   NQS  4.6     urm hold   0.49   0.00      0      0 target memory usage reached
hertr    0   NQS  5.9     urm hold  16.00   0.00    360      0 target memory usage reached
```

The `-w` option adds the following columns to the `ustat` output:

Column	Description
`ReqMem`	Displays the amount of requested memory. This information is supplied by NQS (`qsub`(1)) for batch jobs; system defaults are used for other jobs.
`UseMem`	Displays the actual amount of memory used (smoothed by URM, if smoothing is enabled).
`ReqCpu`	Displays the amount of CPU time requested, in minutes.
`UseCpu`	Displays the actual amount of CPU time used, in minutes.

SEE ALSO

`qsub`(1), `rmgr`(1)

UNICOS Resource Administration, Cray Research publication SG–2302

NAME

uucp – System-to-system copy

SYNOPSIS

uucp [-cCdfjmr] [-n *user*] *source-file...* *destination-file*

IMPLEMENTATION

All Cray Research systems

STANDARDS

XPG4

DESCRIPTION

The uucp utility copies files named by the *source-file* arguments to the *destination-file* argument. The files named can be on local or remote systems.

The uucp utility supports the following options and operands:

-c Does not copy local files to the spool directory for transfer to the remote machine (default).

-C Forces the copy of local files to the spool directory for transfer.

-d Makes all necessary directories for the file copy (default).

-f Does not make intermediate directories for the file copy.

-j Writes the job identification string to standard output. This job identification can be used by uustat(1) to obtain the status or terminate a job.

-m Sends mail to the requester when the copy is completed.

-n *user* Notifies *user* on the remote system that a file was sent.

-r Does not start the file transfer; just queues the job.

destination-file
source-file A path name of a file to be copied to, or from, respectively. Either name can be a path name on the local machine, or can have the form:

system-name!*pathname*

where *system-name* is taken from a list of system names that uucp knows about. The destination *system-name* can also be a list of names such as:

system-name!*system-name*!...!*system-name*!*pathname*

in which case, an attempt is made to send the file via the specified route to the destination. Care should be taken to ensure that intermediate nodes in the route are willing to forward information.

The shell pattern matching notation characters `?`, `*`, and `[. . .]` appearing in *pathname* will be expanded on the appropriate system.

Path names can be one of the following:

- An obsolete path name.
- A path name preceded by *~user* where *user* is a login name on the specified system and is replaced by that user's login directory. Note that if an invalid login is specified, the default is to the public directory (called `PUBDIR`; the actual location of `PUBDIR` is implementation-specific).
- A path name preceded by *~/destination* where *destination* is appended to `PUBDIR`.

  Note: This destination will be treated as a file name unless more than one file is being transferred by this request or the destination is already a directory. To ensure that it is a directory, follow the destination with a `/`. For example, `~/dan/` as the destination will make the directory `PUBDIR/dan` if it does not exist and put the requested files in that directory.
- Anything else is prefixed by the current directory.

If the result is an erroneous path name for the remote system, the copy will fail. If the *destination-file* is a directory, the last part of the source-file name is used.

The read, write, and execute permissions given by `uucp` are implementation-dependent.

The following environment variables affect the execution of `uucp`:

`LANG` Provides a default value for the internationalization variables that are unset or null. If `LANG` is unset or null, the corresponding value from the implementation-specific default locale will be used. If any of the internationalization variables contains an invalid setting, the utility will behave as if none of the variables had been defined.

`LC_ALL` If set to a nonempty string value, overrides the values of all the other internationalization variables.

`LC_COLLATE` Determines the locale for the behavior of ranges, equivalence classes and multicharacter collating elements within bracketed file name patterns.

`LC_CTYPE` Determines the locale for the interpretation of sequences of bytes of text data as characters (for example, single- as opposed to multibyte characters in arguments and input files) and the behavior of character classes within bracketed file name patterns (for example, `'[[:lower:]]*'`).

`LC_MESSAGES` Determines the locale that should be used to affect the format and contents of diagnostic messages written to standard error, and informative messages written to standard output.

LC_TIME	Determines the format of date and time strings output by uucp.
NLSPATH	Determines the location of message catalogs for the processing of LC_MESSAGES.
TZ	Determines the timezone used with date and time strings.

NOTES

The domain of remotely accessible files can (and for obvious security reasons usually should) be severely restricted.

Note that the ! character in addresses has to be escaped when using csh(1) as a command interpreter because of its history substitution syntax. For ksh and sh, the escape is not necessary, but may be used.

As noted above, shell metacharacters appearing in path names are expanded on the appropriate system. On an internationalized system, this is done under the control of local setting of LC_COLLATE and LC_CTYPE. Thus, care should be taken when using bracketed file name patterns, as collation and typing rules may vary from one system to another. Also be aware that certain types of expression (that is, equivalence classes, character classes and collating symbols) need not be supported on non-internationalized systems.

The uucp utility cannot guarantee support for all character encodings in all circumstances. For example, transmission data may be restricted to 7 bits by the underlying network, 8-bit data and file names need not be portable to non-internationalized systems, and so forth. Under these circumstances, it is recommended that only characters defined in the ISO/IEC 646:1991 standard International Reference Version (equivalent to ASCII) 7-bit range of characters be used and that only characters defined in the Portable Filename Character Set be used for naming files.

The uucp utility exists for compliance with the XPG4 standard; therefore, it is a minimal implementation limited to execution only on the current host system.

EXIT STATUS

The following exit values are returned:

0	Successful completion.
>0	An error occurred.

SEE ALSO

mailx(1), uuencode(1), uustat(1), uux(1)

NAME

uuencode, uudecode – Encodes or decodes a binary file

SYNOPSIS

uuencode [*file*] *name*

uudecode [*file* ...]

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The uuencode and uudecode utilities transmit binary files over transmission mediums that support only ASCII data.

The uuencode utility reads *file* (or by default, the standard input) and writes an encoded version to standard output. The encoding uses only printable ASCII characters and includes the mode of the file and the *name* operand for use by uudecode.

The uudecode utility transforms encoded files (or by default, the standard input) into the original form. The file that results is named *name*; it will have the mode of the original file except that *setuid* and execute bits are not retained. The uudecode utility ignores any leading and trailing lines.

The uuencode and uudecode utilities accept the following operands:

file Specifies the name of file to be encoded or decoded.

name Specifies the name of the file that results from decoding.

EXIT STATUS

The uuencode and uudecode utilities exit with one of the following values:

0 Successful completion.

>0 An error occurred.

BUGS

The encoded form of the file is expanded by 35% (3 bytes become 4, plus control information).

EXAMPLES

The following example packages up a source tree, compresses it, encodes it, and mails it to a user on another system. When `uudecode` is run on the target system, the `src_tree.tar.Z` file will be created, which may then be uncompressed and extracted into the original tree.

```
tar cf – src_tree | compress | uuencode src_tree.tar.Z | mail joe
```

SEE ALSO

`compress`(1), `mail`(1)

`uuencode`(5) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

NAME

uustat – uucp status inquiry and job control

SYNOPSIS

uustat [-q]

uustat [-k *jobid*]

uustat [-r *jobid*]

uustat [-s *system*] [-u *user*]

IMPLEMENTATION

All Cray Research systems

STANDARDS

XPG4

DESCRIPTION

The uustat utility displays the status of, or cancels, previously specified uucp requests, or provides general status on uucp connections to other systems. When no options are given, uustat writes to standard output the status of all uucp requests issued by the current user.

The uustat utility supports the following options:

-q Writes the jobs queued for each machine.

-k *jobid* Kills the uucp request whose job identification is *jobid*. The killed uucp request must belong to the person invoking uustat unless that user has appropriate privileges.

-r *jobid* Rejuvenate *jobid*. The files associated with *jobid* are touched so that the modification time is et to the current time. This prevents the cleanup program from deleting the job until the jobs modification time reaches the limit imposed by the program.

-s *system* Writes the status of all uucp requests for system *system*.

-u *user* Writes the status of all uucp requests issued by *user*.

The following environment variables affect the execution of uustat:

LANG Provides a default value for the internationalization variables that are unset or null. If LANG is unset or null, the corresponding value from the implementation-specific default locale will be used. If any of the internationalization variables contains an invalid setting, the utility will behave as if none of the variables had been defined.

LC_ALL If set to a nonempty string value, overrides the values of all the other internationalization variables.

LC_CTYPE	Determines the locale for the interpretation of sequences of bytes of text data as characters (for example, single- as opposed to multibyte characters in arguments).
LC_MESSAGES	Determines the locale that should be used to affect the format and contents of diagnostic messages written to standard error, and informative messages written to standard output.
LC_TIME	Determines the format of date and time strings output by uustat.
NLSPATH	Determines the location of message catalogs for the processing of LC_MESSAGES.
TZ	Determines the timezone used with date and time strings.

NOTES

The uustat utility exists for compliance with the XPG4 standard; therefore, it is a minimal implementation limited to execution only on the current host system.

EXIT STATUS

The following exit values are returned:

0 Successful completion.

>0 An error occurred.

SEE ALSO

uucp(1)

NAME

uux – System-to-system copy

SYNOPSIS

uux [-np] *command-string*

uux [-jnpf7] *command-string*

uux [-] [-jn] *command-string*

IMPLEMENTATION

All Cray Research systems

STANDARDS

XPG4

DESCRIPTION

The uux utility will gather zero or more files from various systems, execute a shell pipeline on a specified system, and then send the standard output of the command to a file on a specified system. Only the first command of a pipeline can have a system-name! prefix. All other commands in the pipeline are executed on the system of the first command.

The following restrictions are applicable to the shell pipeline processed by uux:

- In gathering files from different systems, path name expansion is not performed by uux. Thus, a request such as:

  ```
  uux "c89 remsys!~/*.c"
  ```

  would attempt to copy the file named literally *.c to the local system.
- The redirection operators >>, <<, >|, and >& cannot be used.
- The reserved word ! cannot be used at the head of the pipeline to modify the exit status.
- Alias substitution is not performed.

A file name can be specified as for uucp(1); it can be obsolete path name, a path name preceded by ~*name* (which is replaced by the corresponding login directory), a path name specified as ~/*dest* (*dest* is prefixed by the public directory called PUBDIR; the actual location of PUBDIR is implementation-specific), or a simple file name (which is prefixed by uux with the current directory). See uucp(1) for the details.

The execution of commands on remote systems takes place in an execution directory known to the uucp system. All files required for the execution will be put into this directory unless they already reside on that machine. Therefore, the nonlocal file names (without path or machine reference) must be unique within the uux request.

The uux utility will attempt to get all files to the execution system. For files that are output files, the file name must be escaped using parentheses.

The remote system will notify the user by mail if the requested command on the remote system was disallowed or the files were not accessible. This notification can be turned off by the -n option.

The uux utility supports the following options and operands:

-p
- Makes the standard input to uux the standard input to the *command-string*.

-j Writes the job identification string to standard output. This job identification can be used by uustat(1) to obtain the status or terminate a job.

-n Does not notify the user if the command fails.

command-string
A string made up of one or more arguments that are similar to normal command arguments, except that the command and any file names can be prefixed by *system-name!*. A null *system-name* is interpreted as the local system.

The following environment variables affect the execution of uux:

LANG Provides a default value for the internationalization variables that are unset or null. If LANG is unset or null, the corresponding value from the implementation-specific default locale will be used. If any of the internationalization variables contains an invalid setting, the utility will behave as if none of the variables had been defined.

LC_ALL If set to a nonempty string value, overrides the values of all the other internationalization variables.

LC_CTYPE Determines the locale for the interpretation of sequences of bytes of text data as characters (for example, single- as opposed to multibyte characters in arguments).

LC_MESSAGES Determines the locale that should be used to affect the format and contents of diagnostic messages written to standard error.

NLSPATH Determines the location of message catalogs for the processing of LC_MESSAGES.

The standard output is not used unless the -j option is specified; in that case, the job identification string is written to standard output in the following format:

"%s\n", <*jobid*>

Output files are created or written, or both, according to the contents of *command-string*.

If the -n is not used, mail files will be modified following any command or file-access failures on the remote system.

NOTES

For security reasons, many installations limit the list of commands executable on behalf of an incoming request from `uux`. Many sites permit little more than the receipt of mail via `uux`.

Any characters special to the command interpreter should be quoted either by quoting the entire *command-string* or quoting the special characters as individual arguments.

Typical implementations of this utility require a communications line configured to use the XBD specification, Chapter 9, General Terminal Interface, but other communications means may be used. On systems where there are no available communications means (either temporarily or permanently), this utility will write an error message describing the problem and exit with a nonzero exit status.

As noted in `uucp`(1), shell pattern matching notation characters appearing in path names are expanded on the appropriate local system. This is done under the control of local settings of `LC_COLLATE` and `LC_CTYPE`. Thus, care should be taken when using bracketed file name patterns, as collation and typing rules may vary from one system to another. Also be aware that certain types of expression (that is, equivalence classes, character classes and collating symbols) need not be supported on non-internationalized systems.

The `uux` utility cannot guarantee support for all character encodings in all circumstances. For example, transmission data may be restricted to 7 bits by the underlying network, 8-bit data and file names need not be portable to non-internationalized systems, and so forth. Under these circumstances, it is recommended that only characters defined in the ISO/IEC 646:1991 standard International Reference Version (equivalent to ASCII) 7-bit range of characters be used and that only characters defined in the Portable Filename Character Set be used for naming files.

The `uux` utility exists for compliance with the XPG4 standard; therefore, it is a minimal implementation limited to execution only on the current host system.

EXIT STATUS

The following exit values are returned:

0 Successful completion.

>0 An error occurred.

EXAMPLES

Example 1: The following command gets *file1* from system `a` and *file2* file from system `b`, executes `diff` on the local system, and puts the results in `file.diff` in the local `PUBDIR` directory. (`PUBDIR` is the `uucp` public directory on the local system.)

```
uus "!diff a!/usr/file1 b!/a4/file2 >!~/file.diff"
```

Example 2: The following command will fail because `uux` places all files copied to a system in the same working directory. Although the files `xyz` are from two different systems, their file names are the same and will conflict.

```
uux "!diff a!/usr1/xyz b!/user2/xyz >!~/xyz.diff"
```

Example 3: The following command will succeed (assuming `diff` is permitted on system `a`) because the file local to system `a` is not copied to the working directory, and hence does not conflict with the file from system `c`:

```
uux "a!diff a!/usr/xyz c!/usr/xyz >!~/xyz.diff"
```

SEE ALSO

uucp(1), uuencode(1), uustat(1)

NAME

`val` – Validates SCCS file

SYNOPSIS

```
val -
val [-m name] [-r SID] [-s] [-y type] files
```

IMPLEMENTATION

All Cray Research systems

STANDARDS

XPG4

DESCRIPTION

The `val` utility determines whether the specified *files* are Source Code Control System (SCCS) files meeting the characteristics specified by the option list. Options to `val` may appear in any order. The options consist of keyletter arguments, which begin with a `-` symbol, and named files.

The `-` argument causes reading of the standard input until an end-of-file condition is detected. Each line read is independently processed, as if it were a command line argument list.

The `val` utility generates diagnostic messages on the standard output for each command line and file processed, and it also returns a single 8-bit code upon exit.

The options are defined as follows. The effects of any option apply independently to each named file on the command line.

`-m` *name* The argument value *name* is compared with the SCCS `man/man1/val.1` keyword in *files*.

`-r` *SID* An SCCS delta number. A check is made to determine whether the source identifier (SID) is ambiguous (such as `r1 1` is ambiguous because it physically does not exist, but it implies `1.1`, `1.2`, and so on, which may exist) or invalid (such as, `r1.0` or `r1.1.0` are invalid because neither case can exist as a valid delta number). If the *SID* is valid and not ambiguous, a check is made to determine if it actually exists.

`-s` Silences the diagnostic message normally generated on the standard output for any error that is detected while processing each named file on a given command line.

`-y` *type* The argument value *type* is compared with the SCCS `%Y%` keyword in *files*.

files Specifies the files to be validated.

EXIT STATUS

The 8-bit code returned by `val` is a disjunction of the possible errors; that is, it can be interpreted as a bit string with set bits interpreted as follows (moving from left to right):

Bit 0 = Missing file argument
Bit 1 = Unknown or duplicate keyletter argument
Bit 2 = Corrupted SCCS file
Bit 3 = Cannot open file or file is not SCCS
Bit 4 = *SID* is invalid or ambiguous
Bit 5 = *SID* does not exist
Bit 6 = `%Y%`, `-y` mismatch
Bit 7 = `%M%`, `-m` mismatch

Note that `val` can process two or more files on a given command line, and in turn can process multiple command lines (when reading the standard input). In these cases, an aggregate code is returned - a logical `OR` of the codes generated for each command line and file processed.

MESSAGES

Messages from the `val` utility do not contain the SCCS help codes. The messages are meant to be self-explanatory.

SEE ALSO

`admin`(1), `cdc`(1), `comb`(1), `delta`(1), `get`(1), `help`(1), `prs`(1), `rmdel`(1), `sact`(1), `sccsdiff`(1), `unget`(1), `vc`(1), `what`(1)

`sccsfile`(5) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

NAME

vc – SCCS version control

SYNOPSIS

vc [-a] [-c *char*] [-s] [-t] [*args*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The vc utility copies lines from the standard input to the standard output under control of its arguments and control statements encountered in the standard input. In the process of performing the copy operation, user-declared *keywords* may be replaced by their string *value* when they appear in plain text and/or control statements.

The copying of lines from the standard input to the standard output is conditional, based on tests (in control statements) of keyword values specified in control statements or as vc utility arguments.

A control statement is a single line beginning with a control character, except as modified by the -t option. The default control character is colon (:), except as modified by the -c option. Input lines beginning with a backslash (\), followed by a control character are not control lines and are copied to the standard output with the backslash removed. Lines beginning with a backslash followed by a noncontrol character are copied in their entirety.

A keyword is composed of 9 or fewer alphanumeric characters; the first character must be alphabetic. A value is any ASCII string that can be created with ed(1); a numeric value is an unsigned string of digits. Keyword values may not contain blanks or tabs.

Replacement of keywords by values is done whenever a keyword surrounded by control characters is encountered on a version control statement. The -a option forces replacement of keywords in all lines of text. An uninterpreted control character may be included in a value by preceding it with a \ symbol. If a literal \ is desired, it too must be preceded by a \ symbol.

The vc utility accepts the following options:

-a Forces replacement of keywords surrounded by control characters with their assigned value in all text lines and not just in vc statements.

-c *char* Specifies a control character to be used in place of the colon (:).

-s Silences warning messages (not error messages) that are usually printed on the diagnostic output.

-t Ignores all characters from the beginning of a line up to and including the first tab character for the purpose of detecting a control statement. If one is found, discards all characters up to and including the tab.

args *args* takes the form [*keyword=value* . . . *keyword=value*].

Version Control Statements

`:dcl` *keyword*[, . . ., *keyword*]
Declares keywords. All keywords must be declared.

`:asg` *keyword=value*
Assigns values to keywords. An `asg` statement overrides the assignment for the corresponding keyword on the `vc` command line and all previous `asg` statements for that keyword. Keywords declared, but not assigned values, have null values.

`:if` *condition*
.
.
.
`:end`
Skips lines of the standard input. If *condition* is true, all lines between the `if` statement and the matching `end` statement are copied to the standard output. If *condition* is false, all intervening lines are discarded, including control statements. Intervening `if` statements and matching `end` statements are recognized solely for the purpose of maintaining the proper `if-end` matching.

The syntax of *condition* is as follows:

```
<cond>      ::= [ "not" ] <or>
<or>        ::= <and> | <and> "|" <or>
<and>       ::= <exp> | <exp> "&" <and>
<exp>       ::= "(" <or> ")" | <value> <op> <value>
<op>        ::= "=" | "!=" | "<" | ">"
<value>     ::= <arbitrary ASCII string> | <numeric string>
```

The available operators and their meanings are as follows:

`=`	Equal
`!=`	Not equal
`&`	And
`|`	Or
`>`	Greater than
`<`	Less than
`( )`	Used for logical groupings
`not`	May only occur immediately after the *if*, and, when present, inverts the value of the entire condition

The `>` and `<` operate on only unsigned integer values (such as, `: 012 > 12` is false). All other operators take strings as arguments (such as, `: 012 != 12` is true). The precedence of the operators (from highest to lowest) is as follows:

`= != > <`	all of equal precedence
`&`	
`|`	

Parentheses may be used to alter the order of precedence.

Values must be separated from operators or parentheses by at least one blank or tab.

`::text` Used for keyword replacement on lines that are copied to the standard output. The two leading control characters are removed, and keywords surrounded by control characters in text are replaced by their value before the line is copied to the output file. This action is independent of the `-a` option.

`:on`
`:off` Turns on or off keyword replacement on all lines.

`:ctl` *char* Changes the control character to *char*.

`:msg` *message* Prints the given message on the diagnostic output.

`:err` *message* Prints the given message followed by the following on the diagnostic output:

```
ERROR: err statement on line ... (915)
```

The `vc` utility halts execution, and returns an exit code of 1.

EXIT STATUS

The `vc` utility returns an exit code of 0 on normal termination and returns a 1 if any error occurs.

MESSAGES

Use `help`(1) for explanations.

SEE ALSO

`admin`(1), `cdc`(1), `comb`(1), `delta`(1), `ed`(1), `get`(1), `help`(1), `prs`(1), `rmdel`(1), `sact`(1), `sccsdiff`(1), `unget`(1), `val`(1), `what`(1)

`sccsfile`(5) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

NAME

vi, view, vedit – Invokes the screen-oriented (visual) display editor

SYNOPSIS

vi [-C] [-r] [-R] [-l] [-L] [-t *tag*] [-V] [-w *n*] [-x] [-c *command*] [*file* ...]

Obsolescent version:

vi [-C] [-r] [-R] [-l] [-L] [-t *tag*] [-V] [-w *n*] [-x] [+*command*] [*file* ...]

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4
AT&T extensions (-C, -L, -V, and -x options)

DESCRIPTION

The vi utilty may be involved as view or vedit. The vi (visual) editor is a display-oriented text editor based on an underlying line editor (ex(1)). You can use the command mode of ex(1) from within vi and vice versa.

When using vi, changes you make to the file are reflected on your terminal screen. The position of the cursor on the screen indicates the position within the file.

The vi editor accepts the following options:

-c *command*
+*command* (Obsolescent)
: Interprets the specified ex(1) command-mode *command* before editing begins. When used with the -t option, the command is executed after moving to the tag.

-C : Same as -x but assume *file* began in encrypted form.

-l : Set lisp mode.

-L : List filenames that were saved due to an editor or system crash.

-r : Recovers *file* after an editor or system crash. If you omit *file*, a list of all saved files is printed.

-R : Read-only mode; the readonly flag is set, preventing accidental overwriting of the file.

-t *tag* : Edits the file that contains the *tag* and positions the editor at its definition.

-V : Verbose. Any non-tty input will be echoed on standard error.

-w *n* : Sets the default window size to *n*. This is useful when using the editor over a slow-speed line.

`-x` Encrypts the file as it is written and requires an encryption key that allows it to be read (see `crypt`(1)). See the WARNINGS section for more information.

The `view` utility is the same as `vi`, except that the `readonly` flag is set.

The `vedit` utility is intended for beginners. The `report` flag is set to 1, and the `showmode` and `novice` flags are set. These defaults make it easier to get started learning the editor.

vi Modes

At any one time, you are in one of the following modes in `vi`:

Command — Normal and initial mode. Other modes return to command mode on completion. To cancel a partial command, press the <ESCAPE> key.

Input — Entered by typing `a i A I o O c C s S R`. Text can then be entered. Input mode is normally terminated with the <ESCAPE> key or abnormally with an interrupt.

Last line — Reading input for `: / ?` or `!`; terminate by pressing the <RETURN> key to execute, and interrupt to cancel.

Command Summary

The following is a summary of commands `vi` accepts.

Sample Commands

← ↓ ↑ →

`h j k l`	Same as arrow keys
`i`*text*<ESCAPE>	Inserts *text*
`cw`*new*<ESCAPE>	Changes word to *new*
`ea`*s*<ESCAPE>	Changes word to plural form
`x`	Deletes a character
`dw`	Deletes a word
`dd`	Deletes a line
`3dd`	Deletes three lines
`u`	Undoes previous change to buffer
`ZZ`	Exits `vi`, saving changes
`:q!`<RETURN>	Quits, discarding changes
`/`*text*<RETURN>	Searches for *text*
<CONTROL-u>	Scrolls up
<CONTROL-d>	Scrolls down
`:`*ex cmd*<CR>	Enters any `ex`(1) or `ed`(1) command

Numbers Before vi Commands

You can type numbers as a prefix to some commands. They are interpreted as follows:

Column/line number	`z G`
Scroll amount	`^D ^U`
Repeat effect	Most commands accept numbers to indicate how many times to repeat

Interrupting and Canceling

<ESCAPE>	Ends insert or incomplete command
DELETE	(Delete or rubout) interrupts
^L	Reprints screen if DEL scrambles it
^R	Reprints screen if <CONTROL-l> is → key

File Manipulation

:w<CR>	Writes changes
:q<CR>	Quits
:q!<CR>	Quits, discard changes
:e *name*<CR>	Edits file *name*
:e!<CR>	Reedits, discarding changes
:e + *name*<CR>	Edits, starting at end
:e +*n*<CR>	Edits starting at line *n*
:e #<CR>	Edits alternate file
^6	Synonym for :e #
:w *name*<CR>	Writes file *name*
:w! *name*<CR>	Overwrites file *name*
:sh<CR>	Runs shell; uses exit to return
:!*cmd*<CR>	Runs *cmd*, then returns
:n<CR>	Edits next file in arglist
:n *args*<CR>	Specifies new arglist
^G	Shows current file and line
:ta *tag*<CR>	Positions cursor at tag and saves previous position on the tag stack (see ctags(1))
^]	:ta, following word is *tag*
:pop<CR>	Returns to previous saved position on the tag stack
^T	Same as :pop command

In general, any ex(1) or ed(1) command (such as substitute or global) may be typed, preceded by a colon (:) and followed by a carriage return (<CR>).

Positioning Within File

^F	Forward screen
^B	Backward screen
^D	Scrolls down half screen
^U	Scrolls up half screen
G	Goes to specified line (default last line)
/*pat*	Next line matching *pat*
?*pat*	Previous line matching *pat*
n	Repeats last / or ?
N	Reverses last / or ?
/*pat*/+*n*	*n*th line after *pat*
?*pat*?-*n*	*n*th line before *pat*
]]	Next section/function
[[	Previous section/function

(	Beginning of sentence
)	End of sentence
{	Beginning of paragraph
}	End of paragraph
%	Finds matching ( ) { or }

Adjusting the Screen

^L	Clears and redraws screen
^R	Retypes, eliminating @ lines
z<CR>	Redraws, current line at window top
z-<CR>	Redraws, current line at bottom of window
z .<CR>	Redraws, current line at center of window
/*pat*/z-<CR>	Redraws *pat* line at bottom
z*n* .<CR>	Uses *n*-line window
^E	Scrolls window down a line
^Y	Scrolls window up a line

Marking and Returning

``	Moves cursor to previous context
´´	Moves cursor to first nonwhite in line
m*x*	Marks current position with letter *x*
`*x*	Moves cursor to mark *x*
´*x*	Moves cursor to first nonwhite in line marked with *x*

Line Positioning

H	Goes to top line on screen
L	Goes to last line on screen
M	Goes to middle line on screen
+	Goes to next line, at first nonwhite
-	Goes to previous line, at first nonwhite
<CR>	Same as +
↓ or j	Goes to next line, same column
↑ or k	Goes to previous line, same column

Character Positioning

^	Goes to first nonwhite character
0	Goes to beginning of line
$	Goes to end of line
h or →	Goes backward
l or ←	Goes forward
^H	Same as ←
space	Same as →
f*x*	Finds *x* forward in line
F*x*	Finds f backward in line
t*x*	Up to *x* forward in line

T*x*	Back up to *x* in line
;	Repeats last f F t or T
,	Inverse of ;
|	Goes to specified column
%	Finds matching ({) or }

Words, Sentences, and Paragraphs

w	Word forward
b	Word backward
e	End of word
)	To next sentence
}	To next paragraph
(	Back sentence
{	Back paragraph
W	Blank delimited word
B	Back W
E	To end of W

Corrections during Insert

^H	Erases last character
^W	Erases last word
erase	Your erase, same as ^H
kill	Your kill, erase input this line
\	Quotes ^H, your erase and kill
<ESC>	Ends insertion, goes back to command
DEL	Interrupts, terminates insertion
^D	Uses backtab over *autoindent*
↑^D	Kills *autoindent*, saves for next
0^D	Same as ↑^D, but at margin next also
^V	Quotes nonprinting character
^T	Inserts shiftwidth spaces

Insert and Replace

a	Appends after cursor
i	Inserts before cursor
A	Appends at end of line
I	Inserts before first non blank character
o	Opens line below current line
O	Opens line above current line
r*x*	Replaces single character with *x*
R*text*<ESC>	Replaces characters

Operators

Operators are followed by a cursor motion and affect all text that would have been moved over. For example, because `w` moves over a word, `dw` deletes the word that would be moved over. Double the operator (for example, `dd`) to affect whole lines.

`d`	Deletes
`c`	Changes
`y`	Yanks lines to buffer
`<`	Left shifts
`>`	Right shifts
`!`	Filters through command
`=`	Indents for LISP

Miscellaneous Operations

`C`	Changes rest of line (`c$`)
`D`	Deletes rest of line (`d$`)
`s`	Substitutes characters (`cl`)
`S`	Substitutes lines (`cc`)
`J`	Joins lines
`x`	Deletes characters (`dl`)
`X`	Deletes characters before cursor (`dh`)
`Y`	Yanks lines (`yy`)

Yank and Put

The put operator inserts the text most recently deleted or yanked; however, if a buffer is specified, the text in that buffer is inserted instead.

`p`	Puts text after cursor
`P`	Puts text before cursor
`"`*x*`p`	Puts text from buffer *x*
`"`*x*`y`	Yanks text to buffer *x*
`"`*x*`d`	Puts text into buffer *x*

Undo, Redo, and Retrieve

`u`	Undoes last change to buffer
`U`	Restores current line
`.`	Repeats last change
`"`*d*`p`	Retrieves *d*'th last delete

Regular Expressions

The `vi` editor recognizes the following regular expressions:

[*cccc*]	Matches any of the specified characters (*cccc*)
[^*cccc*]	Matches any character (*cccc*) except those specified
[*c1*-*c2*]	Matches any character in the specified range (*c1* through *c2*)
^	Matches the beginning of a line

^*cccc*	Matches lines that begin with characters *cccc*
$	Matches the end of a line
cccc$	Matches lines that end with characters *cccc*
.	Matches any one character
*	Matches zero or more occurrences of the preceding character
.*	Matches any number of characters
\{#\}	Matches # occurrences of preceding search string
\{#,\}	Matches at least # occurrences of preceding search string
\{*#1*,*#2*\}	Matches between *#1* and *#2* occurrences of preceding search string
\(*string*\)\#	Matches *string* followed by # occurrences of *string*

To match special characters ($. * [ ] ^ \), precede the special character with a backslash (\).

Set Command and .exrc File

You can set defaults for a variety of editing characteristics while editing with vi. To set a characteristic for a particular editing session, enter the set command as follows:

:set [*option* [=*value*]]

(Specific set options follow.)

If you want to set new defaults for all editing sessions, insert the appropriate set command (without the leading :) into a file called .exrc. The editor searches for .exrc in your home directory first and executes the commands in that file. Then it searches for the .exrc file in your current directory and executes the commands in it; the .exrc file is skipped if you do not own the file. (See exrc(5).)

You can also use the EXINIT environment variable to set these defaults; if this environment variable is set, the $HOME/.exrc file will not be processed, but the .exrc file in your current directory will be processed.

To display set options you have changed, enter the set command without options. :set all shows the state of all set options. :set *x* enables the *x* option. :set no*x* disables the *x* option. :set *x*=*value* gives the *x* option the value of *value*. :set *x*? shows the value of the *x* option.

The set options are as follows (the alias, if one exists, is in parentheses):

autoindent (ai)	Continues previous indentation
autoprint (ap)	Prints current line after buffer changed
autowrite (aw)	Writes before changing files
beautify (bf)	Discards all control characters other than tab, new line, and form feed
directory (dir)	Specifies the directory
edcompatible (ed)	Causes suffixes to be remembered

errorbells (eb)	Sounds bell when error occurs
exrc (ex)	Allows execution of .exrc files that reside outside usr's home directory
flash	Flashes screen on errors
hardtabs	Value of any hardware tab settings
ignorecase (ic)	Ignores case when scanning
lisp	() {} are s-expressions
list	Displays ^I for tab, $ at end of line
magic	Turns on the normal metacharacter meaning of ., [, *
mesg	Allows non-vi messages to appear on screen during vi
modelines (ml)	Prevents accidental interpretation of ex: and vi: in files
novice	Makes it easier to learn to use vi
number (nu)	Numbers lines
optimize (opt)	Abolishes carriage returns at the end of lines when printing multiple lines (speeds output on dumb terminals when printing lines with leading white space)
paragraphs (para)	Macro names that start...
posix92 (px92)	Editing behavior is that which conforms to the P1003.2 standard
prompt	When set, prompted with a ":"
readonly	Disallows writing buffer contents to current file name
redraw	Redraws the screen
remap	Macro translation
report	Sets the threshold for the number of lines modified
scroll	Command mode lines
sections (sect)	Macro names...
shell	Shell executed when doing :! or !
shiftwidth (sw)	Gives the width of a software tab stop used in reverse tabbing
showmatch (sm)	Shows the match to) and { when typed
showmode (smd)	Shows insert mode in vi
slowopen (slow)	Stops updates during insert
tabstop (ts)	Software tab stops
taglength	Tag names (labels) only significant to this many characters

`tags`	List of files to be searched by the `:tag` command
`term`	Type of terminal you are using
`terse`	Error messages are shorter
`timeout`	Must enter a macroname in less than one second
`ttytype` (`tty`)	Sets terminal type (same as `set term`)
`warn`	Warning given when `:!` is issued
`window`	Visual mode lines
`wrapscan` (`ws`)	Searches that use regular expressions will wrap around past EOF
`wrapmargin` (`wm`)	Automatically splits line *n* characters from right (breaks at white space)
`writeany` (`wa`)	Allows a write to any file

CAUTIONS

If your terminal definition is not supported through `TERMINFO=/usr/lib/terminfo`, you may need to define your terminal.

WARNINGS

Inclusion of the Data Encryption Standard (DES) encryption code requires a special license for sites outside the United States and Canada. If these encryption functions are not available on your system, check with your system administrator or site analyst.

Tampering with entries in `/usr/lib/terminfo/?/*` (for example, changing or removing an entry) can affect programs such as `vi` that expect the entry to be present and correct. In particular, removing the dumb terminal can cause unexpected problems.

EXIT STATUS

The `vi` utility exits with one of the following values:

0 Successful completion.

>0 An error occurred.

BUGS

Software tabs in insert mode using `^T` work only immediately after the `autoindent`.

On intelligent terminals, left and right shifts do not make use of insert and delete character operations in the terminal.

Minor display glitches can occur under certain circumstances, such as screen wrapping while overwriting characters with the `R` command.

The `ye` sequence yanks a word into the buffer, but it often misses the last character.

FILES

`/usr/lib/terminfo/?/*` Default terminal information database

SEE ALSO

`awk`(1) `crypt`(1), `ctags`(1), `ed`(1), `edit`(1), `ex`(1), `grep`(1), `sed`(1)

`chown`(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

`exrc`(5), `term`(5), `terminfo`(5) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

Learning the vi Editor, Linda Lamb, O'Reilly & Associates, Inc., 1990

NAME

`wait` – Waits for completion of a process

SYNOPSIS

`wait` [*pid*]

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The `wait` utility instructs the system to wait for your background process that has a process ID of *n* and to report its termination status.

If you omit *n*, the system waits for all of the invoking shell's currently active background processes, and exists with an exit status of 0.

The *pid* operand is one of the following:

- An unsigned decimal integer process ID of a command for which `wait` is to wait for the termination.
- A job control job ID that identifies a background process group to be waited for. This notation applies only to invocations of `wait` in the current shell execution environment.

The shell itself executes `wait`, without creating a new process.

The known process IDs apply only for invocations of `wait` in the current shell execution environment.

NOTES

The `wait` utility is a built-in utility to the standard shell (`sh`(1)). An executable version of this utility is available in `/usr/bin/wait`.

The `csh`(1) utility has a built-in `wait` utility with slightly different characteristics. See `csh`(1).

EXIT STATUS

If one or more operands were specified, all of them have terminated or were not known by the invoking shell, and the status of the last operand specified is known, then the exit status of `wait` is the exit status information of the command indicated by the last operand specified. If the process terminated abnormally due to the receipt of a signal, the exit status is greater than 128 and is distinct from the exit status generated by the other signals. Otherwise, the `wait` utility exits with one of the following values:

0 The `wait` utility was invoked with no operands and all process IDs known by the invoking shell have terminated.

1-126 The `wait` utility detected an error.

127 The command identified by the last *pid* operand specified is unknown.

SEE ALSO

`csh`(1), `sh`(1)

NAME

wc – Counts bytes, characters, lines, and words in a file

SYNOPSIS

```
wc [-c] [-l] [-w] [files]
wc -m [-l] [-w] [files]
```

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The wc utility counts bytes, characters, lines, and words in the specified files or in the standard input if you omit *files*. It also keeps a total count for all specified files. *word* is a maximal string of characters delimited by <space>s, <tab>s, or <newline> characters.

The wc utility accepts the following options:

-c Writes to the standard output the number of bytes in each input file.

-l Writes to the standard output the number of <newline> characters in each input file.

-m Writes to the standard output the number of characters in each input file.

-w Writes to the standard output the number of words in each input file.

files Specifies the files whose content are to be counted.

You can use the -l and -w options in any combination with the -c or -m option to specify that a subset of lines and words will be reported along with bytes or characters, respectively. By default bytes, words, and lines are printed.

When you specify *files* on the command line, they are printed along with the counts.

In addition to being useful for determining the size of a text file, wc can also be useful for determining the number of items displayed by other UNICOS commands.

NOTES

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
system, secadm	In a privileged administrator shell environment, shell-redirected I/O is not subject to file protections.

`sysadm` Shell-redirected output is subject to security label restrictions.

If the `PRIV_SU` configuration option is enabled, shell-redirected I/O on behalf of the super user is not subject to file protections.

Because multibyte characters are not supported in the current UNICOS release, the `-m` option assumes 1 byte per character.

EXIT STATUS

The `wc` utility exits with one of the following values:

0 Successful completion.

>0 An error occurred.

EXAMPLES

Example 1: In this example, the output of `who`(1) is piped to `wc`. `who` shows who is logged in to the system, displaying one line per user. By counting the number of lines, `wc` gives a count of how many people are logged into the system. The first number displayed by `wc` is line count, which corresponds to the number of users logged in.

```
who | wc
```

Example 2: A more sophisticated example follows:

```
who | wc | awk '{print "Number of Users:",$1}'
```

NAME

what – Identifies SCCS files and UNICOS Source Manager (USM) files

SYNOPSIS

what [-s] *files*

IMPLEMENTATION

All Cray Research systems

STANDARDS

XPG4

DESCRIPTION

The what utility searches the specified files for all occurrences of the pattern that get(1) substitutes for %Z% (this is @(#)) and prints out what follows until the first ~, >, new-line, \, or null character. If no *files* are specified, what searches standard input.

The what utility is intended to be used in conjunction with the Source Code Control System (SCCS) utility get(1), which automatically inserts identifying information, but it can also be used where the information is inserted manually.

The what utility accepts the following option:

-s Quits after finding the first occurrence of pattern in each file.

files Specifies the files to be identified.

EXIT STATUS

The what utility exits with one of the following values:

0 Matches were found.

>0 An error occurred.

BUGS

It is possible that an unintended occurrence of the pattern @(#) could be found, but this usually causes no harm.

EXAMPLES

If the C program in file `f.c` contains

```
char ident[] = "@(#)identification information";
```

and `f.c` is compiled to yield `f.o` and `a.out`, the command

```
$ what f.c f.o a.out
```

will write

```
f.c:
        identification information
        ...

f.o:
        identification information
        ...

a.out:
        identification information
        ...
```

SEE ALSO

`admin`(1), `cdc`(1), `comb`(1), `delta`(1), `get`(1), `help`(1), `prs`(1), `rmdel`(1), `sact`(1), `sccsdiff`(1), `unget`(1), `val`(1), `vc`(1)

`sccsfile`(5) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

NAME

whatis – Displays a one-line summary about a command

SYNOPSIS

whatis *command* ...

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The whatis utility displays the header line of *command* from the manual section. You can then run the man(1) command to get more information. If the line starts with *name*(*section*) ..., you can enter man *section name* to display the documentation for it.

The function of the whatis utility is identical to the -f option of the man(1) command.

The whatis utility displays information from the /usr/man/whatis file.

FILES

/usr/man/whatis Database

SEE ALSO

apropos(1), man(1)

NAME

whereis – Locates source, binary, and/or manual for program

SYNOPSIS

/usr/ucb/whereis [-s] [-b] [-m] [-u] [[-S *dirs*] [-B *dirs*] [-M *dirs*] -f] *names*

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The whereis utility locates source, binary, and/or manual sections for specified files. The supplied names are first stripped of leading path name components and any (single) trailing extension of the form *file*.*ext* (for example .c). Prefixes of s. that result from use in source code control are also dealt with. The whereis utility then attempts to locate the desired program in a list of standard places.

The whereis utility accepts the following options:

-s Searches only for sources.

-b Searches only for binaries.

-m Searches only for manual sections.

-u Searches for unusual entries. A file is said to be unusual if it does not have one entry of each requested type. Thus, whereis -m -u * requests those files in the current directory that have no documentation.

-S *dirs* Changes or otherwise limits the places where whereis searches for sources to *dirs*.

-B *dirs* Changes or otherwise limits the places where whereis searches for binaries to *dirs*.

-M *dirs* Changes or otherwise limits the places where whereis searches for manual sections to *dirs*.

-f Terminates the last such directory list and signals the start of file names.

names Specifies file names to be located.

BUGS

Because the program uses chdir(2) to run faster, path names specified with -M, -S, and -B must be full path names; that is, they must begin with a /.

EXAMPLES

The following commands find all the files in /usr/bin that are not documented in /usr/man/cat1 with source in /usr/src/cmd:

```
cd /usr/bin
whereis -u -m -M /usr/man/cat1 -S /usr/src/cmd -f *
```

FILES

`/usr/src/*`	Source code for commands
`/usr/man/*`	Text files for manuals
`/lib`, `/etc`, `/bin`, `/usr/bin`, `/usr/ucb`, `/usr/lbin`	Executable binaries of commands

SEE ALSO

`man`(1)

`chdir`(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

NAME

whichcat – Returns the name of the message system catalog being accessed

SYNOPSIS

whichcat [-l] *group* [*groups*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The whichcat utility returns the path name that contains the message system catalog for the group code(s) *group*. This command verifies that the expected catalog is being referenced or helps determine why no catalog is found.

If no options are specified, only the path name where the catalog is found is returned.

The whichcat utility accepts the following options and arguments:

-l Lists the paths that whichcat searched in looking for the catalogs for *group*.

group Group code of the message system catalog whose path is returned by whichcat. At least one group must be specified. Multiple groups can be specified.

NOTES

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
system, secadm	In a privileged administrator shell environment, allowed to write shell-redirected output to any file.
sysadm	Shell-redirected output is subject to security label restrictions.

If the PRIV_SU configuration option is enabled, the super user can write shell-redirected output to any file.

SEE ALSO

caterr(1), catxt(1), explain(1), gencat(1)

catgetmsg(3C), catgets(3C), catmsgfmt(3C), catopen(3C) in the *UNICOS System Libraries Reference Manual*, Cray Research publication SR–2080

nl_types(5), msg(7D) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

Cray Message System Programmer's Guide, Cray Research publication SG–2121

NAME

who – Reports who is on the system

SYNOPSIS

who -s [-A] [-b] [-h] [-H] [-l] [-m] [-p] [-r] [-t] [-T] [-u] *file*

who [-a] [-A] [-b] [-d] [-h] [-H] [-l] [-m] [-p] [-r] [-t] [-T] [-u] *file*

who [-q] [-n *number*] *file*

who am i

who am I

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4
AT&T extensions (-A, -d, and -n options)

DESCRIPTION

The who utility lists the user name, terminal line, login time, elapsed time since activity occurred on the line, and the process ID of the command interpreter (shell) for each current UNICOS user. It examines the /etc/utmp file to obtain its information. If you omit *file*, that file (which must be in utmp(5) format) is examined instead. Usually, *file* will be /etc/wtmp, which contains a history of all the logins since the file was last created.

The who utility with the -m or the am i operands identifies the invoking user.

Except for the default -s option, the general format for output entries is as follows:

name [*state*] *line time activity pid* [*comment*] [*exit*]

The *name* is the user's login name. The *state* describes whether someone else can write to that terminal. If the terminal can be written by anyone, a + appears. A - appears if it is not. root can write to all lines having a + or a - in the *state* field. If a bad line is encountered, a ? is printed. The *line* is the name of the line as found in the directory /dev. The *time* is the time that the user logged in. The *activity* is the number of hours and minutes since activity last occurred on that particular line. A dot (.) indicates that the terminal has seen activity in the last minute and is therefore ''current''. If more than 24 hours have elapsed or the line has not been used since boot time, the entry is marked old. This field is useful when trying to determine whether a person is working at the terminal. The *pid* is the process ID of the user's shell. The *comment* is the comment field associated with this line as found in /etc/inittab (see inittab(5)). This can contain commentary information. If no comment exists in /etc/inittab, the network host identifier of each user is displayed in parentheses under the comment field; otherwise, it appears after the comment field.

With options, who can list logins, logoffs, reboots, and changes to the system clock, as well as other processes spawned by the init process.

The who utility accepts the following options and operands:

-a Processes /etc/utmp or the specified *file* with all options turned on.

-A Displays utmp entries marked as accounting information.

-b Indicates the time and date of the last reboot of the system under the *line* and *time* headings, respectively.

-d Displays all processes that have expired and not been respawned by init. The *exit* field appears for dead processes and contains the termination and exit values (as returned by wait(2)) of the dead process. This can be useful in determining why a process terminated.

-h Displays network host identifiers.

-H Displays a header.

-l Lists only those lines at which the system is waiting for someone to log in. The *name* field is LOGIN in such cases. Other fields are the same as for user entries, except that the *state* field does not exist.

-m Lists information only about the current terminal.

-n *number*
Specifies the *number* of users per line for -q. The -q option spaces out the users names by using a format similar to that used by the ls(1) utility.

-p Lists any other process that is currently active and has been previously spawned by init. The *name* field is the name of the program executed by init as found in /etc/inittab. The *state, line,* and *activity* fields have no meaning. The *comment* field shows the *id* field of the line from /etc/inittab that spawned this process. See inittab(5).

-q Displays only the names and the number of users currently logged on; this is a quick who. When you use this option, all other options except -n are ignored.

-r Indicates the current *run-level* of the init process. In addition, this option produces the process termination status, process ID, and process exit status under the *idle, pid,* and *comment* headings, respectively.

-s Is the default and lists only the *name, line,* and *time* fields.

-t Indicates the last change to the system clock (through the date(1) utility) by root. See su(1).

-T Displays the *state* of the terminal line.

-u Lists information about users currently logged on the system.

file Specifies the file to contain login information.

NOTES

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
`system`, `secadm`	In a privileged administrator shell environment, shell-redirected I/O is not subject to file protections.
`sysadm`	Shell-redirected output is subject to security label restrictions.

If the `PRIV_SU` configuration option is enabled, shell-redirected I/O on behalf of the super user is not subject to file protections.

EXIT STATUS

The `who` utility exits with one of the following values:

0 Successful completion.

>0 An error occurred.

FILES

`/etc/inittab`

`/etc/utmp`

`/etc/wtmp`

SEE ALSO

`date`(1), `login`(1), `ls`(1), `mesg`(1), `su`(1)

`wait`(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

`inittab`(5), `utmp`(5) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

`init`(8) in the *UNICOS Administrator Commands Reference Manual*, Cray Research publication SR–2022

NAME

whoami – Displays the effective current user name

SYNOPSIS

```
/usr/ucb/whoami
```

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The whoami utility displays the login name corresponding to the current effective user ID. If you have used su(1) to temporarily adopt another user, whoami will report the login name associated with that user ID. whoami gets its information from the geteuid(2) system call and the getpwuid(3C) library routine.

FILES

/etc/passwd	User name data base

SEE ALSO

logname(1), su(1), who(1)

getuid(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

getpwuid(3C) in the *UNICOS System Libraries Reference Manual*, Cray Research publication SR–2080

NAME

`write` – Lets you write to another user

SYNOPSIS

`write` *user_name* [*terminal*]

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The `write` utility copies lines from your terminal to that of another user. When first called, it sends the following message to the other person:

```
Message from
yourname
(tty##)
[date]...
```

When it has completed the connection successfully, it sends two alert sequences to your own terminal, as well as to the recipient's terminal, to indicate what you are typing is being sent.

The message recipient should write back at this point. Communication continues until an end of file is read from the terminal or an interrupt is sent. At that point, the `write` utility writes `<EOT>` on the other terminal and exits.

The `write` utility accepts the following operands:

user_name The login name of person to whom the message will be written.

terminal Indicates terminal to which to send.

If you want to write to a user who is logged in more than once, use the *terminal* argument to indicate which terminal to send (such as, `ttyp001`); otherwise, the first writable instance of *user_name* found in `/etc/utmp` is assumed and the following message is posted:

```
user_name is logged on more than one place.
You are connected to "terminal".
Other locations are:
terminal
```

To deny or grant permission to write to another user's terminal, use the `mesg`(1) utility. By default, writing to others is usually allowed. Certain utilities, in particular `pr`(1), inhibit messages to prevent interference with their output. However, if the user has super-user permissions, messages can be forced onto a write-inhibited terminal.

If the character `!` is found at the beginning of a line, the `write` utility calls the shell to execute the rest of the line as a command.

The `write` utility detects nonprintable characters before sending them to the recipient's terminal. Control characters appear as a `^` character followed by the appropriate ASCII character. Characters with the high-order bit set will appear in meta-notation (for example, `'\0372'` is displayed as `'M-z'`).

The following protocol is suggested for using `write`: when you first `write` to another user, wait for them to `write` back before starting to send a line. Each person should end a message with a distinctive signal (that is, `(o)` for ''over'') so that the other person knows when to reply. The signal `(oo)` (for ''over and out'') is suggested when conversation is to be terminated.

NOTES

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
`system`, `secadm`	Allowed to write to any user.
`sysadm`	Allowed to write to any user, subject to security label restrictions on the user's terminal path. Shell-redirected I/O is subject to security label restrictions.

If the `PRIV_SU` configuration option is enabled, the super user is allowed to write to any user.

EXIT STATUS

The `write` utility exits with one of the following values:

0 Successful completion.

>0 The addressed user is not logged on, or the addressed user denies permission.

MESSAGES

`User is not logged on.`
The person to whom you are trying to `write` is not logged on.

`Permission denied.`
The person to whom you are trying to `write` denies that permission (with `mesg`(1)).

`Warning: You have your terminal set to "mesg n". No reply possible.`
Your terminal is set to `mesg n` and the recipient cannot respond to you.

`Can no longer write to user.`
The recipient has denied permission (`mesg n`) after you had started writing.

`User is not at terminal.`
The recipient is not logged on at the specific terminal.

FILES

`/etc/utmp` To find *user_name*

SEE ALSO

`mail`(1), `mesg`(1), `pr`(1), `talk`(1B), `who`(1)

General UNICOS System Administration, Cray Research publication SG–2301

NAME

xargs – Constructs argument lists and executes a utility

SYNOPSIS

xargs [-p] [-t] [-e[*eofstr*] | -E *eofstr*] [-i[*replstr*] | -I *replstr*] [-l[*number*] | -L *number*] [-s *size*] [-x] [*utility* [*arguments*]]

xargs [-p] [-t] [-e[*eofstr*] | -E *eofstr*] [-i[*replstr*] | -I *replstr*] [-n *nargs*] [-s *size*] [-x] [*utility* [*arguments*]]

IMPLEMENTATION

Cray PVP systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The xargs utility combines the fixed *arguments* with arguments read from standard input to execute *utility* one or more times. The options specified determine the number of arguments read for each *utility* invocation and the manner in which they are combined.

The *utility* argument, which may be a shell file, is searched for using the user's $PATH. If you omit *utility*, /bin/echo is used.

Arguments read in from standard input are defined to be contiguous strings of characters delimited by one or more <blank>, <tab>, or <newline> characters; empty lines are always deleted. You can embed <blank>s and <tab> characters as part of an argument if escaped or quoted; characters enclosed in quotation marks (single or double) are taken literally, and the delimiting quotation marks are removed. Outside of quoted strings, a backslash (\) escapes the next character.

Each argument list is constructed starting with the *arguments*, followed by some number of arguments read from standard input (exception: see -I flag). Options -I, -i, -L, -l, and -n determine how arguments are selected for each command invocation. When none of these options are specified, the *arguments* are followed by arguments read continuously from standard input until an internal buffer is full, and then *utility* is executed with the accumulated arguments. This process is repeated until all arguments have been read. When option conflicts exist (such as -l with -n, or -e with -E), the last flag has precedence.

The xargs utility accepts the following options:

-E *eofstr* Use *eofstr* as the logical EOF string. Underscore (_) is assumed for the logical EOF string if neither -e nor -E is specified. The xargs utility reads standard input until either end-of-file or the logical EOF string is encountered.

-e[*eofstr*] This option is equivalent to -E *eofstr*. If the option-argument is not specified, the logical EOF string capability is disabled and underscores are taken literally.

`-I` *replstr* Insert mode. The *utility* will be executed for each line from standard input, taking the entire line as a single argument, inserting it in *arguments* for each occurrence of *replstr*. A maximum of five arguments in *arguments* can each contain one or more instances of *replstr*. Any `<blank>` or a `<tab>` characters at the beginning of each line are ignored. Constructed arguments may not be larger than {`NAME_MAX`} characters. Option `-x` is forced when this option is used.

`-i`[*replstr*] This option is equivalent to `-I` *replstr*. If the option-argument is not specified, the string `{}` is assumed for *replstr*.

`-L` *number* The *utility* will be executed for each nonempty *number* lines of arguments from standard input. The last invocation of *utility* will be with fewer lines of arguments if fewer than *number* remain. A line is considered to end with the first `<newline>`, unless the last character of the line is a `<blank>` or a `<tab>`; a trailing `<blank>` or `<tab>` signals continuation through the next nonempty line. Option `-x` is forced when this option is used.

`-l`[*number*] This option is equivalent to `-L` *number*. If the option-argument is not specified, 1 is assumed.

`-n` *nargs* Executes *utility* by using as many standard input arguments as possible, up to *nargs* arguments maximum. Fewer arguments will be used if their total size is greater than *size* characters, and for the last invocation if there are fewer than *nargs* arguments remaining. If you also specify the `-x` option, each *nargs* arguments must fit in the *size* limitation; otherwise, `xargs` terminates execution.

`-p` Prompt mode. The user is asked whether to execute *utility* each invocation. Trace mode (`-t`) is turned on to print the command instance to be executed, followed by a `?...` prompt. A reply of `y` (optionally followed by anything) executes the command; anything else, including just a `<carriage-return>`, skips that particular invocation of *utility*.

`-s` *size* The maximum total size of each argument list is set to *size* characters. *size* must be a positive integer less than or equal to 3996. If you omit `-s`, 3996 is the default. The character count for *size* includes one extra character for each argument and the count of characters in the command name.

`-t` Trace mode. The *utility* and each constructed argument list are `echo`ed to standard error just prior to their execution.

`-x` Causes `xargs` to terminate whether any argument list is greater than *size* characters. The `-x` option is forced by the `-I`, `-i`, `-L`, and `-l` options. When you do not specify `-I`, `-i`, `-L`, `-l`, or `-n` options, the total length of all arguments must be within the *size* limit.

The `xargs` utility terminates if *utility* terminates because of a signal, a system call failure while trying to execute *utility*, or if it cannot execute *utility*. When *utility* is a shell program, it should explicitly `exit` (see `sh`(1)) with an appropriate value to avoid accidentally returning with -1.

EXIT STATUS

The `xargs` utility exits with one of the following values:

0 All invocations of *utility* returned exit status 0.

1–125 A command line that meets the specified requirements could not be assembled, one or more of the invocations of *utility* returned a nonzero exit status, or some other error occurred.

126 The utility specified by *utility* was found, but it cannot be invoked.

127 The utility specified by *utility* cannot be found.

EXAMPLES

Example 1: The following command line moves all files from directory `$1` to directory `$2`, and `echo` each `mv`(1) utility just before doing it:

```
ls $1 | xargs -i{} -t mv $1/{} $2/{}
```

Example 2: The following combines the output of the parenthesized commands onto one line, which is then echoed to the end of file `log`:

```
(logname; date; echo $0 $*) | xargs >>log
```

Example 3: The user is asked which files in the current directory will be archived and archives them into file `arch` one at a time, or archives them into `arch` many files at a time.

```
ls | xargs -p -l1 ar r arch
ls | xargs -p -l1 | xargs ar r arch
```

Example 4: The following command line executes `diff`(1) with successive pairs of arguments originally typed as shell arguments:

```
echo $* | xargs -n 2 diff
```

SEE ALSO

`echo`(1), `find`(1), `sh`(1)

NAME

yacc – Yet another compiler compiler

SYNOPSIS

yacc [-b *file_prefix*] [-d] [-l] [-p *sym_prefix*] [-s *scale*] [-t] [-v] *file*

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The yacc utility converts a context-free grammar found in *file* into a set of tables for a simple automaton that executes an LR(1) parsing algorithm (see NOTES section for a definition of LR(1)). The grammar may be ambiguous; specified precedence rules are used to break ambiguities. The yacc utility produces the y.tab.c file as output.

The output file, y.tab.c, must be compiled by the C compiler to produce the yyparse program. This program must be loaded with the lexical analyzer program, yylex, as well as main and yyerror (an error-handling routine). These routines must be supplied by the user; lex(1) is useful for creating lexical analyzers usable by yacc.

Run-time debugging code is always generated in y.tab.c under conditional compilation control. By default, this code is not included when y.tab.c is compiled.

The yacc utility accepts the following options:

-b *file_prefix* Uses *file_prefix* instead of y as the prefix for all output filenames. The code file y.tab.c, the header file y.tab.h (created when -d is specified), and the description file y.output (created when -v is specified), are changed to *file_prefix*.tab.c, *file_prefix*.tab.h, and *file_prefix*.output, respectively.

-d Generates a y.tab.h file with the #define statements. that associate the token codes assigned by yacc with the user-declared token names. This allows source files other than y.tab.c to access the token codes.

-l Prevents the use of #line constructs in y.tab.c. This should be used only after the grammar and the associated actions are fully debugged.

-p *sym_prefix* Uses *sym_prefix* instead of yy as the prefix for all external names produced by yacc. The names affected include the functions yyparse(), yylex(), and yyerror(), and the variables yylval, yychar, and yydebug.

`-s` *scale*	Changes the default sizes of arrays that `yacc` uses to compile the grammar (productions, states, nonterminals, and so on) up or down according to scale. The scale must be a positive real number. To increase the array size, use a number greater than 1; to decrease the array size use a number less than 1.
`-t`	Changes the default to include run-time debugging code when `y.tab.c` is compiled. Whether or not the `-t` option was used, the run-time debugging code is under the control of `YYDEBUG`, a preprocessor symbol. When `YYDEBUG` has a nonzero value, the debugging code is included. When its value is 0, the code is not included. The size and execution time of a program produced without the run-time debugging code is smaller and slightly faster.
`-v`	Prepares the `y.output` file. This contains a description of the parsing tables and a report on conflicts generated by ambiguities in the grammar.

NOTES

Because file names are fixed, only one `yacc` process can be active in a directory.

`LR`(1) is a type of Deterministic Context-Free Language (DCFL). `LR`(1) stands for "left-to-right scan of the input producing a rightmost derivation and using 1 symbol of lookahead on the input." `LR`(1) grammars have great importance for compiler design because they are broad enough to have efficient parsers that are essentially Deterministic Pushdown Automata (DPDA).

EXIT STATUS

The `yacc` utility exits with one of the following values:

0 Successful completion.

>0 An error occurred.

MESSAGES

The number of reduce-reduce and shift-reduce conflicts is reported on the standard error output; a more detailed report is found in the `y.output` file. Similarly, if some rules cannot be reached from the start symbol, this will also be reported.

FILES

file_prefix`.output`	Temporary files
file_prefix`.tab.c`	Temporary files
file_prefix`.tab.h`	Defines token names
`yacc.tmp`	Temporary files
`yacc.debug`	Temporary files
`yacc.acts`	Temporary files
`/usr/lib/yaccpar`	Parser prototype for C programs

SEE ALSO

`lex`(1)

lex & yacc, Doug Brown and Tony Mason, O'Reilly & Associates, Inc., 1992.

The UNIX Programming Environment, Brian W. Kernighan and Rob Pike, Prentice-Hall, Inc., 1984.

NAME

ypcat – Prints values in a network information service (NIS) database

SYNOPSIS

```
/usr/bin/ypcat [-k] [-t] [-d domainname] mname
/etc/yp/ypcat -x
```

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The ypcat utility prints out values in a network information service (NIS) map. Because ypcat uses the NIS, no NIS server is specified.

The ypcat utility accepts the following options:

-k Displays the keys for those maps in which the values are null or in which the key is not part of the value. (None of the maps derived from files that have an ASCII version in /etc fall into this class.)

-t Inhibits translation of *mname* to map name. For example, ypcat -t passwd fails because no map is named passwd, whereas ypcat passwd is translated to ypcat passwd.byname.

-d *domainname*
Specifies a domain other than the default domain. *domainname* returns the default domain.

-x Displays the map nickname table. It lists the nicknames the utility knows, and it indicates the map name associated with each nickname.

mname Specifies a map name or a map nickname.

To look at the network-wide password database, passwd.byname (with the nickname *passwd*), enter the following:

```
ypcat passwd
```

See ypfiles(5) and ypserv(8) for an overview of the NIS.

SEE ALSO

domainname(1), ypmatch(1)

ypfiles(5) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

ypserv(8) in the *UNICOS Administrator Commands Reference Manual*, Cray Research publication SR–2022

NAME

ypmatch – Prints the value of one or more keys from a network information service (NIS) map

SYNOPSIS

```
/usr/bin/ypmatch [-d domain] [-k] [-t] keys... mname
/etc/yp/ypmatch -x
```

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The ypmatch utility prints the values associated with one or more keys from an NIS map (database) specified by *mname*, which can be either a map name or a map nickname.

You can specify multiple *keys*; the same map is searched for all. The capitalization and length of keys must be exact. No pattern matching is available. If a key is not matched, a diagnostic message is produced.

The ypmatch utility accepts the following options:

-d *domain*	Specifies a domain other than the default domain.
-k	Prints the key itself, followed by a colon (:), before printing the value of a key. This is useful only if the keys are not duplicated in the values, or you have specified so many keys that the output could be confusing.
-t	Inhibits translation of nickname to map name. (For example, ypmatch -t zippy passwd fails because there is no map named passwd, while ypmatch zippy passwd is translated to ypmatch zippy passwd.byname.)
keys...	Specifies the name of the key or keys for which ymatch prints a value.
mname	Specifies the name or nickname of the NIS map that contains the key.
-x	Displays the map nickname table. It lists the nicknames the utility knows, and it indicates the map name associated with each nickname.

SEE ALSO

ypcat(1)

ypfiles(5) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

NAME

yppasswd – Changes login password in network information service (NIS)

SYNOPSIS

/usr/bin/yppasswd [*name*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The yppasswd command changes (or installs) a password associated with the user *name* (your own name by default) in the NIS. The NIS password may be different than the one on your own machine.

The yppasswd command prompts for the old NIS password and then for the new one. The caller must supply both. The new password must be typed twice, to prevent mistakes.

New passwords must consist of at least 4 characters if they use a sufficiently rich alphabet and at least 6 characters if monocase.

Only the owner of the name or the super user can change a password; in either case, you must prove you know the old password.

BUGS

The update protocol passes all the information to the server in one RPC call, without ever looking at it. Thus, if you type in your old password incorrectly, you are not notified until after you have entered your new password.

SEE ALSO

passwd(1)

ypfiles(5) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

yppasswdd(8) in the *UNICOS Administrator Commands Reference Manual*, Cray Research publication SR–2022

NAME

ypwhich – Specifies which host is the network information service (NIS) server or map master

SYNOPSIS

```
/usr/bin/ypwhich [-d domain] [-V1 | -V2] [hostname]
/usr/bin/ypwhich [-t] [-d domain] -m [mname]
/usr/bin/ypwhich -x
```

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The ypwhich utility specifies which NIS server supplies NIS services to an NIS client, or which is the master for a map. If invoked without arguments, it gives the NIS server for the local machine. If you specify *hostname*, that machine is queried to find out which NIS master it is using.

The ypwhich utility accepts the following options:

-d *domain*	Specifies *domain* instead of the default domain.
-V1	Specifies which server is serving v.1 NIS protocol-speaking client processes.
-V2	Specifies which server is serving v.2 NIS protocol client processes. If neither version is specified, ypwhich attempts to locate the server that supplies the (current) v.2 services. If no v.2 server currently is bound, ypwhich attempts to locate the server that supplies the v.1 services. Because NIS servers and NIS clients are both backward compatible, users seldom must be concerned about which version is currently in use.
hostname	Specifies the name of the machine to query.
-t	Inhibits nickname translation. It is useful if a map name is identical to a nickname. This is not true of any Sun-supplied map.
-m [*mname*]	Finds the master NIS server for the map specified by *mname*. You cannot specify *hostname* with -m. The *mname* argument can be a map name or a nickname for a map. When you omit *mname*, -m produces a list of available maps.
-x	Displays the map nickname table. It lists the nicknames the utility knows, and it indicates the map name associated with each nickname.

See ypfiles(5) and ypserv(8) for an overview of the NIS.

SEE ALSO

`ypfiles`(5) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

`rpcinfo`(8), `ypserv`(8), `ypset`(8) in the *UNICOS Administrator Commands Reference Manual*, Cray Research publication SR–2022

NAME

zcat – Displays expanded files

SYNOPSIS

zcat [*filename*]

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4
AT&T

DESCRIPTION

The zcat utility writes to standard output the uncompressed form of files that have been compressed using the compress(1) utility. It is the equivalent of uncompress -c. Input files are not affected.

The zcat utility accepts the following operand:

filename The pathname of a file previously processed by the compress(1) utility. If *filename* already has the .Z suffix specified, it is used as submitted. Otherwise, the .Z suffix is appended to the file name prior to processing.

EXIT STATUS

The following exit values are returned:

0 Successful completion.

>0 An error occurred.

SEE ALSO

compress(1), uncompress(1)

''A Technique for High Performance Data Compression,'' Terry A. Welch, *IEEE Computer*, vol. 17, no. 6 (June 1984), pp. 8–19.