UNIX® Station User's Guide

SU–0107 C

Cray Research, Inc.

Requests for copies of Cray Research, Inc. publications should be sent to the following address:

Cray Research, Inc.
Distribution Center
2360 Pilot Knob Road
Mendota Heights, MN 55120

NEW FEATURES

The UNIX Station User's Guide includes one new command and several new aliases. The **strstat** command provides status information for data streams. The **exit** and **quit** commands are new aliases for the **end** command, and the **stat** command is a new alias for the **status** command.

RECORD OF REVISION CRAY RESEARCH, INC. PUBLICATION NUMBER SU-0107

Each time this manual is revised and reprinted, all changes issued against the previous version are incorporated into the new version and the new version is assigned an alphabetic level.

Every page changed by a reprint with revision has the revision level in the lower righthand corner. Changes to part of a page are noted by a change bar in the margin directly opposite the change. A change bar in the margin opposite the page number indicates that the entire page is new. If the manual is rewritten, the revision level changes but the manual does not contain change bars.

Requests for copies of Cray Research, Inc. publications should be directed to the Distribution Center and comments about these publications should be directed to:

CRAY RESEARCH, INC.
1345 Northland Drive
Mendota Heights, Minnesota 55120

Revision	Description
	September 1985 - Original printing.
A	April 1986 - Rewrite. This printing reorganizes the manual and adds support of UCB UNIX 4.2BSD. Miscellaneous technical and editorial changes have also been made to bring the documentation into agreement with version 2.0 of the station. This printing obsoletes all previous printings.
B	May 1987 - Rewrite. This printing adds support of the CRAY-2 computer system and the Cray operating system UNICOS. Miscellaneous technical and editorial changes have also been made to bring the documentation into agreement with version 3.00 of the station. All trademarks are now documented in the record of revision.
C	June 1988 - Reprint with revision for release 3.02 of the station. The **strstat**, **exit**, **quit**, and **stat** commands were added. The **scpqsub**(1) command was changed to **uscpqsub**(1). A description of **qsub**(1) was added.

PREFACE

This publication introduces users to the software that logically links the UNIX operating system to the Cray operating system UNICOS or COS on a Cray computer system. The software providing the logical link is a set of programs running under the UNIX operating system.

This software is referred to as the *UNIX station* throughout this publication. The UNIX station requires either UNICOS version 2.0 or higher or COS version 1.13 or higher, and selected operating systems based on and including UNIX System V Release 2.0 and the Fourth Berkeley Software Distribution (4.2BSD) under license from The Regents of the University of California.†

This publication is intended for users of UNICOS and COS who want to transfer jobs, data, and interactive commands to, from, or through a connected UNIX operating system. It provides reference information, tutorial information for new users of the UNIX station, and an overview of COS concepts. (See the UNICOS Primer, publication SG-2010, for an overview of UNICOS concepts.)

It is assumed that readers of this publication have a general understanding of the characteristics of UNIX and Cray operating systems. The following Cray Research, Inc. (CRI) publications also support the UNIX station:

- UNIX Station Command Reference Manual, publication SU-0105
- UNIX Station Administrator's Guide, publication SU-0106

Familiarity with the following publications is recommended, depending on the operating systems used. All publication are CRI publications unless otherwise noted.

CRI publications:

- COS Reference Manual, publication SR-0011
- COS Message Manual, publication SR-0039
- UNICOS Primer, publication SG-2010
- UNICOS User Commands Reference Manual, publication SR-2011
- UNICOS Kernel Error Message Manual, publication SR-2015
- Text Editor (TEDI) User's Guide, publication SG-0055

† Contact your local CRI sales office for information about specific UNIX operating system/hardware implementation.

AT&T publications:

- UNIX System V Release 2.0 User Reference Manual DEC Processors, publication 307-109
- UNIX System V Release 2.0 User Guide, publication 307-100

Hewlett-Packard publications:

- HP-UX Reference Manual, part 09000-90009

Pyramid Technology Corporation publications:

- OSx 4.2BSD Manual Pages, publication 4100-0046
- OSx System V Manual Pages, publication 4100-0047
- OSx 4.0 Release Bulletin, publication 4100-0048

Silicon Graphics, Inc. publications:

- IRIS User's Guide, Version 2.1, publication 007-1101-020
- IRIS Series 3000 Owner's Guide, Version 1.1, publication 007-5220-010
- Getting Started With Your IRIS Workstation, publication 007-7001-010

Sun Microsystems, Inc. publications:

- Beginner's Guide to the Sun Workstation, part 800-1169-01
- Commands Reference Manual for the Sun Workstation, part 800-1172-001

CONVENTIONS

This manual uses the following conventions:

Convention	Description
bold	Indicates all literal strings, including command names, directory names, file names, path names, and command options
italic	Indicates variables and words or concepts being defined
underscore	Specifies the minimum number of characters required for the station command or option to be recognized
option1|*option2*	Indicates that one of the station command options must be specified

Convention	Description
[] brackets	Enclose optional portions of a command line
... ellipsis	Indicates optional use of the preceding item two or more times in succession
*choice*1 *choice*2	Indicates two or more literal station parameters, only one of which can be specified
command(1)	Refers to a UNICOS command entry in the UNICOS User Commands Reference Manual
man(1) *title*	Displays reference manual pages from the UNICOS User Commands Reference Manual

Unless otherwise specified, this manual uses the term *Cray operating system* to refer to both of the Cray operating systems, UNICOS and COS, and the term *Cray job* to refer to both UNICOS and COS jobs.

Case is not significant under COS. COS job control statements can be entered in lowercase, uppercase, or a combination of both. However, case is significant under UNICOS, and UNICOS commands must be entered in lowercase. Unless otherwise specified in this manual, references to the COS data transfer statements ACQUIRE, FETCH, and DISPOSE, which appear in uppercase, also refer to the equivalent UNICOS commands **acquire**(1), **fetch**(1), and **dispose**(1), which must be entered in lowercase.

READER COMMENTS

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

- Call our Technical Publications department at (612) 681-5729 during normal business hours (Central Time).
- Send us electronic mail from a UNICOS or UNIX system, using one of the following electronic mail addresses:

  ihnp4!cray!publications or **sun!tundra!hall!publications**

- Use the postage-paid Reader's Comment form at the back of this manual.

* Write to us at the following address:

  Cray Research, Inc.
  Technical Publications Department
  1345 Northland Drive
  Mendota Heights, Minnesota 55120

We value your comments and will respond to them promptly.

CONTENTS

1. INTRODUCTION

This section introduces you to the characteristics, capabilities, and configuration of the UNIX station.

If you are a new user of the UNIX station, sample terminal sessions included throughout this manual provide you with on-line experience in preparing, submitting, and monitoring jobs processed by Cray operating systems. If you are a new user of UNICOS, see the UNICOS Primer for a discussion of basic UNICOS concepts. If you are a new user of COS, see appendix A for a discussion of basic COS concepts.

1.1 UNIX STATION DESCRIPTION

The UNIX station is a software product that controls the link between a front-end computer system running the UNIX operating system and a Cray computer system running UNICOS or a Cray computer system (other than a Cray-2 system) running COS.

The UNIX station allows you to access Cray facilities from the UNIX environment. When using the UNIX station you can, for example, submit a job to the Cray operating system for processing. The UNIX station interactive facility allows you to log on directly to the Cray operating system and observe the results of each COS job control statement or UNICOS command before you proceed.

1.2 UNIX STATION CAPABILITIES

The UNIX station allows you to perform the following tasks:

- Submit batch jobs from UNIX front-end terminals
- Transfer files from the front-end computer system to the Cray system
- Transfer datasets or files from the Cray system to the front-end computer system

- Control Cray jobs from UNIX front-end terminals
- Display Cray operating system status from UNIX front-end terminals
- Log on interactively to the Cray operating system
- Redirect interactive input and output
- Access on-line help information for UNIX station commands

1.3 UNIX STATION CONFIGURATION

The physical connection between the Cray system and the UNIX front-end system is provided by a Network Systems Corporation (NSC) HYPERchannel network or a combination of networks, such as HYPERchannel, HYPERbus, and Ethernet networks. Figure 1-1 shows the resulting configuration.

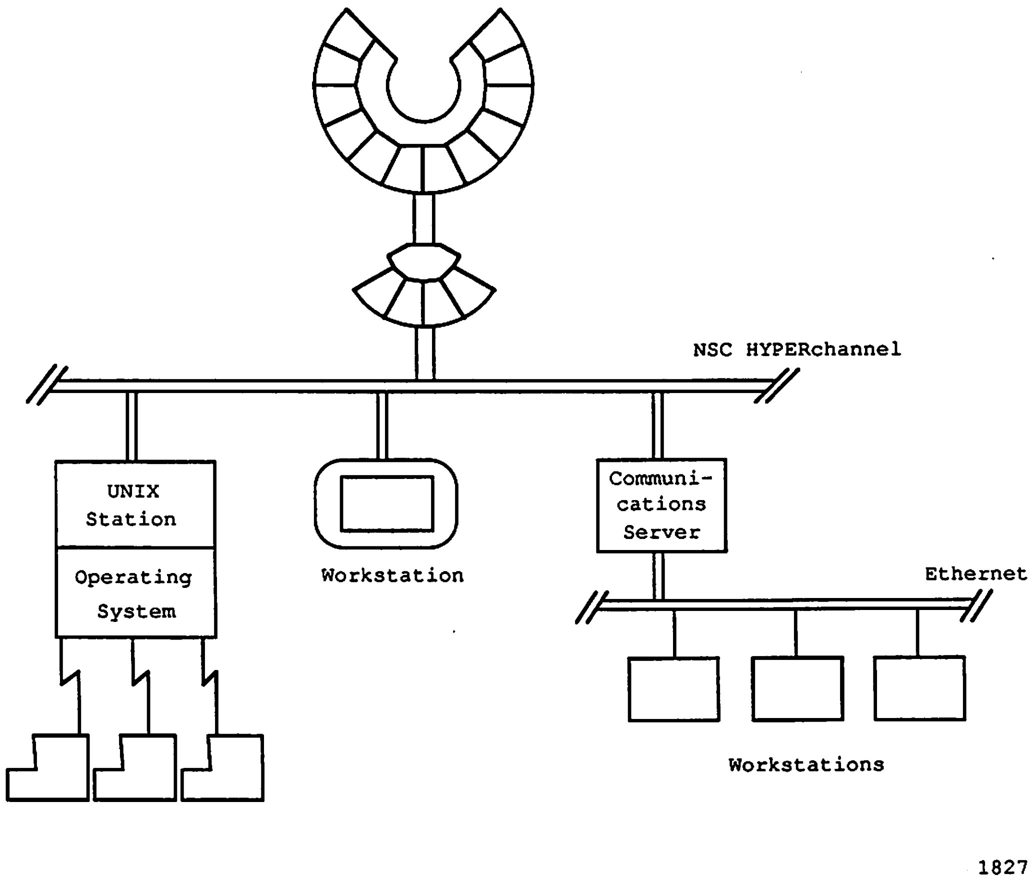

Figure 1-1. UNIX Station Configuration

2. GETTING STARTED

This section provides the basic start-up information you need to use the UNIX station. The following topics are discussed:

- Setting up your environment
- Setting environment variables
- Entering station commands
- Using station command format
- Accessing on-line help information

As a user of the UNIX station you have access to both the batch station and the interactive station. This section primarily discusses the use of COS and UNICOS batch station commands. See section 7 for a discussion of COS interactive facilities or section 8 for a discussion of UNICOS interactive facilities.

2.1 SETTING UP YOUR ENVIRONMENT

The first step in setting up your environment is to ensure that the environment variable $PATH contains a path name for the **cs** and **ias** commands, which allow you to access the batch and interactive stations, respectively. See your system administrator for more information about $PATH.

The second step is to set up your user default output directory. To do this, you can define an environment variable called $CRAYOUT that defines the complete path name of the directory to which your output files will be sent. This path name must not end with a slash (/); the station automatically adds one to your path name. The station then sends your output files to $CRAYOUT as its first choice.

You are not required to define $CRAYOUT because the station sends your output files to the default subdirectory defined in the station's configuration file, which is concatenated to your home directory. For example, if you are user **abc**, the default subdirectory defined in the configuration file is **tmp**, and your home directory is **/usr/abc**, the station uses the path **/usr/abc/tmp** to define your output directory.

See the UNIX Station Administrator's Guide for information on the station default output directory.

If you do not have a subdirectory with the same name as the default subdirectory, or if the station cannot write to your subdirectory, the station designates its own lost and found directory (a path defined in a station configuration file) as the default output directory.

NOTE

Your default output directory is determined at the beginning of each ias session, and remains your default output directory for the duration of that session. With batch processing, however, a default output directory is determined at the time each job is submitted through the station, whether the job is submitted from the shell (snapshot mode) or from the station console (console mode).

2.2 SETTING ENVIRONMENT VARIABLES

Environment variables are used to change the defaults used by the cs and ias commands. The only variable required to be set is $PATH; however, many users choose to set $IASINIT.

$IASINIT points to a file in your home directory that typically contains account information. To start an interactive session, you can enter ias -i to log on to the Cray operating system and automatically send the contents of the file pointed to by $IASINIT to the Cray operating system. This file can contain any job control statement and have any name.

For COS, if accounting is required, the first line of the file must contain a valid COS ACCOUNT job control statement. Subsequent lines can contain the COS job control statements, such as ACCESS or ASSIGN, needed for the interactive session. In UNICOS, the first line of the file must contain a valid UNICOS user login name, and the second line must contain the account password. Contact your system administrator for information on your Cray system login name and password.

The environment variables $CRAY, $STATID, and $CRAYOUT are also available if you want to override the system defaults; however, it is not necessary to do so. Both **cs** and **ias** refer to the $CRAY environment variable when choosing the destination mainframe. If you set this variable, you do not need to specify the **-c** option to override the default defined by your station administrator. This is useful at sites with more than one Cray mainframe. The $STATID environment variable specifies a default station ID. if you set this variable, you do not need to specify the **-s** option. See your system administrator for valid station IDs. The $CRAYOUT environment variable specifies the complete path name of your default output directory. A $CRAYOUT directory must be writeable by the ID running the station daemons.

If you use the Bourne shell, assign values to the variables by entering the following commands in your **.profile** file.

Example:

```
    IASINIT=$HOME/.iasinit
    export IASINIT
    CRAY=mainframeID
    export CRAY
    CRAYOUT=pathname
    export CRAYOUT
    STATID=stationID
    export STATID
```

If you use the C shell, assign values to the variables by entering the following commands in your **.login** file.

Example:

```
    setenv CRAY mainframeID
    setenv IASINIT $HOME/.iasinit
    setenv STATID stationID
    setenv CRAYOUT pathname
```

2.3 ENTERING STATION COMMANDS

You can communicate with the Cray system in several ways by entering station commands associated with the UNIX station. For example, you can use one of the following station commands:

- The **submit** command submits a batch job from your terminal.
- The **status** command monitors the status of the Cray operating system.
- The **save** command stages datasets to Cray mass storage (COS only).

There are two main types of UNIX station commands: batch and interactive. This subsection discusses how to enter batch station commands. Sections 7 and 8 discuss the COS and UNICOS interactive station commands, respectively.

You can enter batch station commands either from the shell (*snapshot mode*) or from the station console (*console mode*). Enter batch station commands after the shell mode prompt (example prompts are $ or %; $ is used as the shell mode prompt in this manual) with the following command format. If you do not specify *command*, you will enter console mode.

```
cs [-c crayid] [-s statid] [command]
```

-c *crayid*
: Destination Cray mainframe at sites with more than one Cray system. If this option is omitted, the destination defined in the environment variable $CRAY is used. If $CRAY is not defined, your station administrator defines the default destination by ST_CRAY in the **st_hafile** configuration file.

-s *statid*
: A 2-character station mainframe identifier; used at sites with multiple stations connected to a Cray mainframe, or sites with test stations. If this option is omitted, the ID defined by the $STATID environment variable is used. If $STATID is not defined, the default station ID is either ST_DEFIDI or ST_DEFIDB, which your station administrator defined in the **st_hafile** configuration file.

command
: Batch station command; some batch user station commands have no effect if entered from snapshot mode. See the UNIX Station Command Reference Manual for more information on these commands. If no command is specified, you will enter console mode.

The following example, entered from snapshot mode, displays the status of COS jobs.

```
$ cs status
JOBNAME  JSQ  DC  STATUS    CLASS    PRI  FL   CPU    LIMIT  MF  TID
-------  ---- --  --------  -----    ---  ---  -----  ------ --  ---
XXXX2     33  IN  EXECUTE   LARGE      5  232    747    1000 V3  U0123
UNIX      19  IN  SUSP-RCY  EXPRESS    8   45      0       8 UX  abc
UNIX      76  IN  WAIT-CPU  EXPRESS    8   71      0       8 UX  abc
                            End of data
```

You can also enter batch station commands from console mode. Console mode allows you to enter station commands after the console mode prompt (>); cs is not entered before each command.

If *command* is not specified, you will enter console mode. The system responds by displaying the batch user station command names at the top of the screen and the console mode prompt (>) at the bottom of the screen. Figure 2-1 shows the default console mode display.

```
CRAY station v.31a2 crayid host    (help)    mm/dd/yy  hh/mm/ss  Frame n

                         STATION v.31a2 COMMANDS
                         ------------------------

     +             -              channel         class          cray
     dataset       device         drop            end            enter
     exit          flush          help            jstat          kill
     limit         link           log             message        operator
     queue         quit           recover         refresh        rerun
     resume        route          save            scroll         shutdown
     stat          status         statclass       storage        stream
     strstat       submit         suspend         switch


     >
```

Figure 2-1. Default Console Mode Display

The header line of the console mode display contains the mainframe destination, station command currently executing, date, time, and frame number.

There are batch user commands that allow you to modify the console mode display screen. For example, the + and - commands scroll the console mode display screen forward and backward, respectively. The **refresh** batch station command controls the time period between each update of the display screen. The **scroll** station command displays the command input and response on the full screen instead of only on the bottom three lines of the screen. For more information on these station commands, see the UNIX Station Command Reference Manual.

You can now enter station commands while the requested display is automatically updated. For example, the status command displays the status of COS jobs in a screen display similar to figure 2-2:

```
CRAY station v. 3.02 crayid host   (status)   mm/dd/yy hh/mm/ss Frame n

NAME     JSQ  DC  STATUS    CLASS   PRI  FL   CPU  LIMIT     MF  TID
------- ---- --  --------  ------- ---  ---- ---  --------  --  -----
RECALL     2  IN  WAIT-EVT  NORMAL    7  114    0  16777215  V3  CRYST
MKDB     544  IN  EXECUTE   NORMAL    7  304  114       250  CX  bjs
MANAGE    56  IN  WAIT-TIM  NORMAL   13  131   53  16777215  V3  CRYST
FDUMP    538  IN  WAIT-EVT  NORMAL    7   83   27        35  SX  sga
X5SYS2   370  PR  WAIT-XFR           13    0    0         0  **  PASS
SW       433  IN  WAIT-SYS  NORMAL   14   55    0  16777215  SN  sw
X5SYS1   523  PR  WAIT-XFR           13    0    0         0  **  PASS
$RECL37  542  IN  WAIT-TIM  NORMAL    7  117    0        60  V3  CRYST
CL       490  IN  WAIT-SYS  NORMAL   14   59    0  16777215  FI  cl


>status
>
```

Figure 2-2. COS Job Status Console Mode Display

To return to the shell, type **end**, **exit**, or **quit** after the console mode prompt.

2.4 USING STATION COMMAND FORMAT

UNIX station commands have the following format:

station-command required-arguments [*optional-arguments*]...

The *station-command* describes the station function that is to be performed. The *required arguments* and *optional arguments* modify the station command by providing additional information on how the command should be executed. An argument may also specify the name of a file.

You can enter a batch station command in uppercase, lowercase, or a combination of both, and abbreviate it to its shortest unique identifier. For example, entering the following command has the same effect as entering **cs status**:

```
$ cs STaTu
```

The following command, however, is not unique because it can refer to the **status** or **statclass** command, and a station message is returned to indicate the error:

```
$ cs Sta
Command "sta" not unique
```

In the UNIX station documentation, the shortest unique identifier for each command is represented by underlining the minimum number of letters you need to enter. The station commands are described in the UNIX Station Command Reference Manual. For example, the first two letters of the scroll command are the minimum number of letters you need to enter for the command to be recognized.

2.5 ACCESSING ON-LINE HELP INFORMATION

The UNIX station provides on-line help for the batch and interactive station commands available on COS. Section 7, Using the COS Interactive Facility, describes the on-line help information for the interactive station commands. Enter **cs help** to display the batch station commands. The following display appears on your screen. This display is also the initial display when you enter console mode.

```
                      STATION v.31a2 COMMANDS
                      -----------------------

+              -              channel        class         cray
dataset        device         drop           end           enter
exit           flush          help           jstat         kill
limit          link           log            message       operator
queue          quit           recover        refresh       rerun
resume         route          save           scroll        shutdown
stat           statclass      status         storage       stream
strstat        submit         suspend        switch
```

To request help for an individual batch station command, enter **help** followed by the station command name, as in the following example:

```
    $ cs help submit
    Help v.3.02 for submit command:

    Send a job to the CRAY input queue

    SUBMIT [-u] path|-

       path = pathname of file to submit
       - = file copied from standard input
       If the -u option is specified, the file will be unlinked
       when the CRAY acknowledges receipt of the file.
```

3. SUBMITTING JOBS TO COS

This section describes how to submit your batch job files from the front-end computer system to COS for processing. The section is divided into the following areas:

- Job submission overview
- Preparing the COS job file
- Submitting and monitoring the COS job
- Managing the COS job output
- Reading the COS job output file

If you are a new user of COS, see appendix A for a discussion of basic COS job control language (JCL). For detailed COS information, see the COS Reference Manual.

3.1 JOB SUBMISSION OVERVIEW

The process by which you prepare, submit, and monitor jobs that are processed by COS is divided into three stages. Figure 3-1 describes this process.

3.2 PREPARING THE COS JOB FILE

The first step in job submission is preparing a COS job file on the front-end computer system. A COS job file contains JCL (a collection of COS job control statements) and optional source code and data in the required COS format.

1. On the front-end computer system, prepare the job file or files that will be submitted to COS. The first element in the job file must be COS JCL, which provides explicit instructions for COS on how to process the job. The job file or files you create on the front-end system can contain other elements, such as code and data to be used by the program. The job file must have other (public) read access.

2. Use the **submit** station command to send the job file or files from the front-end computer system to the Cray system for processing. You can use other station commands to monitor and modify the job as it runs. The COS job output is returned to the front-end computer system and placed in your default output directory.

3. Use the front-end computer system to manage the output from the submitted job. For example, you can read the file at your terminal, print a copy of it, or modify it if it contains errors.

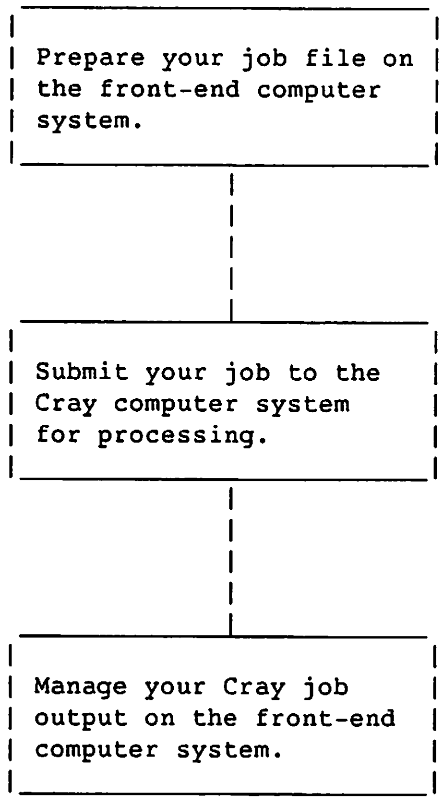
Prepare your job file on the front-end computer system.
Submit your job to the Cray computer system for processing.
Manage your Cray job output on the front-end computer system.

Figure 3-1. Steps in Submitting a COS Job

Figure 3-2 is an example of a COS job deck containing a control statement file, a source file, and a data file. The station supports the COS concept of multiple files within a dataset by allowing an end-of-file record to be placed in a UNIX file. On input to COS, a line with an end-of-file marker (usually ~e, but a different marker can be specified by the station administrator) causes an end-of-file record to be written. (On output from COS, each end-of-file record is translated into an end-of-file marker, except for the last one.)

As with all UNIX files submitted to the Cray system through the station, the COS job file must have other (public) read access.

```
JOB,JN=jobname.                                      }
ACCOUNT,AC=20042510000,US=myid,UPW=mypass.           }  JCL control
CFT77.                                               }  statement file
SEGLDR,GO.                                           }
AUDIT.                                               }
~e                                                   }  End-of-file marker
      J=50                                           }
      DO 10 I=1,10                                   }
      J=J+1                                          }
      PRINT 50,J                                     }
   10 CONTINUE                                       }  Source file
      STOP                                           }
   50 FORMAT(' ',I3)                                 }
      END                                            }
~e                                                   }  End-of-file marker
AAA                                                  }
BBB                                                  }  Optional data file
CCC                                                  }
DDD                                                  }
~e                                                   }  End-of-file marker
```

Figure 3-2. Example of a COS Job Deck

3.3 SUBMITTING AND MONITORING THE COS JOB

After you have prepared your job file, you can submit it to the Cray system. Use the **submit** station command to send your job for COS processing. The format of **submit** is as follows.

```
submit [-u] pathname|-
```

-u	Unlink option; the job file sent to COS is deleted from the front-end system after it has been received and acknowledged by COS. The directory and file must have group and other (public) write access.
pathname	Path name of the COS job file to be executed. This is a required parameter. The *pathname* can begin with a slash (/) to specify a full UNIX path name or a tilde (~) to specify a path beginning with a specified user's home directory. If *pathname* does not begin with a slash or tilde, the path starts in the current directory. If *pathname* is a dash (-), input is taken from **stdin** (standard input).

The following example submits the job file **/usr/abc/cosjob**, regardless of the current directory:

```
    $ cs submit /usr/abc/cosjob
```

To monitor the status of your COS job, enter the **status** station command. To display your own jobs, use the **p** (personal) option, as follows:

```
    $ cs status p
    NAME      JSQ  DC   STATUS     CLASS   PRI   FL    CPU    LIMIT  MF   TID
    -------- ---- --  --------  -------- ---  ---  -----  -------  --  -----
    COSJOB   2111  IN  EXECUTE   MEDIUM   14  117    152      500  UX    abc
```

This status display provides the following information about your COS job: the job name is COSJOB, the job sequence number is 2111, the disposition code is IN (job dataset), the status is EXECUTE (job is executing), the job class assignment is MEDIUM, the job priority is 14, the field length is 117, the CPU time used is 152 seconds, the CPU time limit is 500 seconds, the station ID is UX, and the terminal ID is abc.

The **jstat** station command provides an alternate method for monitoring your COS job. Use the **drop**, **kill**, **suspend**, **resume**, and **rerun** station commands to control your executing job. For more information on these commands, see the UNIX Station Command Reference Manual.

3.4 MANAGING THE COS JOB OUTPUT

After COS processing, the job output file is returned to the front-end computer system to your default output directory (refer to subsection 2.1, Setting Up Your Environment).

The name of the returned job output file consists of a 1-character to 7-character job name (specified by the COS JOB control statement), followed by a period and a unique 6-character extension. To determine which job output file was most recently written to your default output directory, use the UNIX **ls** command with the **-t** option, as in the following example:

```
    $ ls -t
    COSJOB.93487a
    COSJOB.23437a
    COSJOB.20394a
```

The file most recently written to your output directory appears at the top of the listing.

3.5 READING THE COS JOB OUTPUT FILE

After COS has run your job, the output is returned to your default output directory. Use an editor or the UNIX **cat** command to display your job output file.

If you included the AUDIT control statement in your job file, the first page of the COS program output contains a report on permanent datasets stored on the Cray system. By using the available parameters, the AUDIT control statement reports on numerous variables, including the following, which appear from left to right in the report:

- Permanent dataset name
- Dataset identifier
- Edition number
- User identification
- Creation date/time
- Last access date/time
- Last modified date/time
- Last dumped date/time
- Name of the device on which dataset resides

If system defaults are used for the AUDIT statement in the JCL file, the AUDIT report in the job output is similar to that shown in figure 3-3.

```
AUDIT          COS X.14          10/30/84  14:21:50              Page    1

OWN = U9999

--------------------------------------------------------------------------------
PDN              ID         ED                LAST      LAST      LAST      DEVICE
        SZ     RT    ACC TA PAM    CREATED    ACCESSED  MODIFIED  DUMPED
--------------------------------------------------------------------------------

CFTFILE                      1     11/29/84  10/07/84            10/07/84  49-A2-24
        512     45      3 N  N     10:02:17  10:03:23            10:05:45

        1 DATASET,        1 BLOCK,                   602 WORDS
```

1551

Figure 3-3. AUDIT Report

The next page of the program output contains a header line and a listing of your executable Fortran code. The Fortran code is assigned line numbers, as in the following example.

```
      J=50
      DO 10 I=1,10
      J=J+1
      PRINT 50,J
10    CONTINUE
50    FORMAT(' ',I3)
      STOP
      END
```

The results of your program appear on the bottom of the following page, under the heading TOTAL. In this example, the results appear as follows:

```
TOTAL:
 51
 52
 53
 54
 55
 56
 57
 58
 59
 60
```

The log file for your job appears at the end of the job output. The log file is an abbreviated history of the progress of your job through the Cray system. Each job control statement is listed sequentially, followed by a message associated with the job step. The log file also gives the clock time, accumulated CPU time, and COS information for each job step. Figure 3-4 shows an example of a COS job log file. For more information on the job log file, see the COS Reference Manual.

```
10:16:19      0.0000   CSP    --------------------------------------------------------------------------
10:16:19      0.0000   CSP
10:16:19      0.0000   CSP    ------ Use the    NEWS    control statement for general CRAY news ------
10:16:19      0.0001   CSP   ---------------------------------------------------------------------------------
10:16:20      0.0002   CSP          CRAY X-MP SERIAL-201/40    CRI - MENDOTA HEIGHTS, MINN. 08/14/85
10:16:20      0.0002   CSP
10:16:20      0.0002   CSP          CRAY OS - EDITION 292 OF XMs           COS X.15  ASSEMBLY DATE 08/14/85
10:16:20      0.0002   CSP
10:16:20      0.0002   CSP
10:16:20      0.0002   CSP   JOB,JN=SAMPJOB,MFL=70000,T=1.
10:16:20      0.0010   CSP   ACCOUNT,AC=,US=,UPW=.
10:16:57      0.0218   EXP   *
10:16:57      0.0218   EXP   *  GENERATE A PERMANENT DATASET.
10:16:57      0.0218   EXP   *
10:16:57      0.0223   CSP   COPYF,O=PERMDS.
10:16:58      0.0225   USER  I0048 - COPY OF        3 RECORDS       1 FILES COMPLETED
10:16:58      0.0230   CSP   COPYF,O=PERMDS.
10:16:59      0.0232   USER  I0048 - COPY OF        4 RECORDS       1 FILES COMPLETED
10:16:59      0.0234   CSP   SAVE,DN=PERMDS.
10:17:00      0.0234   PDM   PD000 - PDN = PERMDS           ID =           ED =    5  OWN = U1723
10:17:00      0.0234   PDM   PD000 - SAVE    COMPLETE
10:17:00      0.0235   CSP   EXIT.
10:17:00      0.0235   CSP   END OF JOB
10:17:00      0.0235   CSP
10:17:00      0.0235   CSP
10:17:05      0.0236   USER      JOB NAME -                          SAMPJOB
10:17:05      0.0236   USER      USER NUMBER -                       U1723
10:17:05      0.0236   USER      TIME EXECUTING IN CPU -             0000:00:00.0236
10:17:05      0.0237   USER      TIME WAITING TO EXECUTE -           0000:00:26.8951
10:17:05      0.0237   USER      TIME WAITING FOR I/O -              0000:00:19.6661
10:17:05      0.0237   USER      TIME WAITING IN INPUT QUEUE -       0000:00:13.3264
10:17:05      0.0237   USER      MEMORY * CPU TIME (MWDS*SEC) -               0.00111
10:17:05      0.0237   USER      MEMORY * I/O WAIT TIME (MWDS*SEC) -          0.95467
10:17:05      0.0238   USER      MINIMUM JOB SIZE (WORDS) -               28672
10:17:05      0.0238   USER      MAXIMUM JOB SIZE (WORDS) -               53760
10:17:05      0.0238   USER      MINIMUM FL (WORDS) -                     25088
10:17:05      0.0238   USER      MAXIMUM FL (WORDS) -                     50176
10:17:05      0.0238   USER      MINIMUM JTA (WORDS) -                     3584
10:17:05      0.0238   USER      MAXIMUM JTA (WORDS) -                     3584
10:17:05      0.0238   USER      DISK SECTORS MOVED -                       512
10:17:05      0.0238   USER      FSS SECTORS MOVED -                          0
10:17:05      0.0238   USER      USER I/O REQUESTS -                         38
10:17:05      0.0238   USER      USER I/O SUSPENSIONS -                     117
10:17:05      0.0238   USER      OPEN CALLS -                                12
10:17:05      0.0238   USER      CLOSE CALLS -                               14
10:17:05      0.0239   USER      MEMORY RESIDENT DATASETS -                   0
10:17:05      0.0239   USER      TEMPORARY DATASET SECTORS USED -             0
10:17:05      0.0239   USER      PERMANENT DATASET SECTORS ACCESSED -       119
10:17:05      0.0239   USER      PERMANENT DATASET SECTORS SAVED -            1
10:17:05      0.0239   USER      SECTORS RECEIVED FROM FRONT END -            0
10:17:05      0.0239   USER      SECTORS QUEUED TO FRONT END -                0
```

1101-01

Figure 3-4. Example of a COS Job Log File

4. SUBMITTING JOBS TO UNICOS

This section describes how to submit your batch job files from the front-end computer system to UNICOS for processing. The section is divided into the following areas:

- Job submission overview
- Preparing the UNICOS job file
- Submitting and monitoring the UNICOS job
- Managing the UNICOS job output
- Reading the UNICOS job output file

If you are a new user of UNICOS, see the UNICOS Primer for an introduction to UNICOS concepts.

4.1 JOB SUBMISSION OVERVIEW

You can submit batch jobs to UNICOS through use of the Network Queueing System (NQS), which lets you submit, terminate, monitor, and, within limits, control batch jobs submitted to the batch system. You can send batch requests to your own system or to other appropriately configured computer systems in your facility's network. Figure 4-1 describes the process by which you prepare, submit, and monitor jobs processed by UNICOS.

4.2 PREPARING THE UNICOS JOB FILE

First you must create a shell script of commands that executes a sequence of actions for the batch request. Within this file, you can include flags that qualify the **qsub**(1) command, as long as the flags appear before the shell commands and are preceded by the **# QSUB** characters. Section 6, Transferring Data Between UNIX and UNICOS, provides more information about the **qsub** command.

1. On the front-end computer system, prepare the job file or files that will be submitted to UNICOS. The first element in the job file must be the NQS commands that provide explicit instructions for processing the job. The job files you create on the front-end system can contain other elements, such as code and data to be used by the program. The job file must have other (public) read access.

2. Use the **submit** station command to send the job file or files from the front-end computer system to UNICOS for processing. You can use other station commands to monitor and modify the job as it runs. The UNICOS job output is returned to the front-end computer system and placed in your default output directory.

3. Use the front-end computer system to manage the output from the submitted job. For example, you can read the file at your terminal, print a copy of it, or modify it if it contains errors.

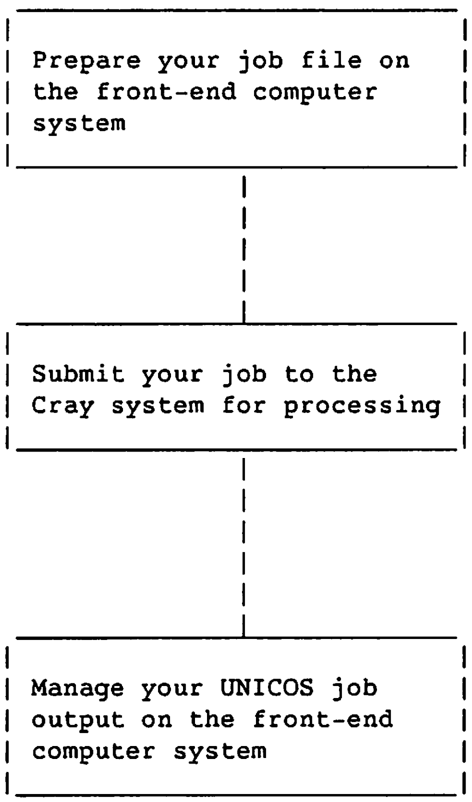
Prepare your job file on the front-end computer system
Submit your job to the Cray system for processing
Manage your UNICOS job output on the front-end computer system

Figure 4-1. Steps in Submitting a UNICOS Job

Figure 4-2 is an example of a UNICOS job file in which an NQS command (qstat) and **qsub** options (-q and -r) are contained in a UNIX file. (The concept of multiple files in a dataset is not supported under UNICOS.)

As with all UNIX files submitted to the Cray system, the UNICOS job file must have other (public) read access.

```
# USER=userid PW=passwd      )
# QSUB-q batch               |
# QSUB-r test.job            }    NQS commands
# QSUB-lT 500                |
qstat -l                     |
rm -f for.f                  )
cat << EOF > for.f           }    I/O redirection statement
        program test         )
        j=50                 |
        do 10 i=1,10         |
        j=j+1                }    Source file
        print 50,j           |
     10 continue             |
        stop                 |
     50 format(' ',i3)       |
        end                  )
EOF                          }    End of input to cat
cft77 -e mx -V for.f         )
segldr for.o                 }    Compile and loader statements
a.out                        )
```

Figure 4-2. Example of a UNICOS Batch Job

4.3 SUBMITTING AND MONITORING THE UNICOS JOB

After you have prepared your job file, you can submit it to the Cray system. Use the **submit** station command to send your job for UNICOS processing. The format of **submit** follows:

submit [**-u**] *pathname*|-

-u Unlink option; the job file sent to UNICOS is deleted from the front-end system after it has been received and acknowledged by UNICOS. The directory and file must have group and other (public) write access.

pathname Path name of the UNICOS job file to be executed. The *pathname* can begin with a slash (/) to specify a full UNIX path name or a tilde (~) to specify a path beginning with a specified user's home directory. If *pathname* does not begin with a slash or tilde (/ or ~), the path starts in the current directory. If *pathname* is a dash (-), input is taken from **stdin** (standard input).

The following example submits the **test.job** file, regardless of the current directory:

```
    $ cs submit /usr/scp/test.job
```

To monitor the status of your UNICOS job, enter the **status** station command using the **p** (personal) operand, as follows:

```
    $ cs status p
    NAME       JSQ  DC  STATUS     CLASS    PRI    FL  CPU    LIMIT   MF  TID
    ------- -----  --  --------   ------- ---  ----  ---  -------   --  ---
    TEST.JO 38667  IN  ended      output   15   117  152      500   UX  abc
```

The **status** command displays only NQS jobs and interactive users communicating through the UNICOS Station Call Processor (USCP). The status display provides the following information on your UNICOS job: the job name is TEST.JO, the job sequence number is 38667, the disposition code is IN (job file), the status is ended (job has executed), the job class assignment is output, the job priority is 15, the field length is 117, the CPU time used is 152 seconds, the limit of CPU time limit is 500 seconds, the station ID is UX, and the terminal ID is abc.

The **jstat** station command provides an alternative method for monitoring your UNICOS job. Use the **drop** and **kill** station commands to control your executing job. For more information on these commands, see the UNIX Station Command Reference Manual.

4.4 MANAGING THE UNICOS JOB OUTPUT

The station returns the job output file to the front-end computer system at a location specified by the $CRAYOUT environment variable or the system default. Refer to subsection 2.1, Setting up Your Environment.

If you did not specify the -r option to the **qsub**(1) command in the job file, the name of the returned job output file consists of a UNICOS user ID, followed by an underscore, a period, and a unique 6-character extension. To determine which job output file was most recently written to your default output directory, use the UNIX **ls** command with the -t option, as in the following example. The file most recently written to your default output directory appears at the top of the listing.

```
    $ ls -t
    test.jo.93487a
    st____.23437a
    stress.20394a
    stress.92348a
    st____.23948a
```

4.5 READING THE UNICOS JOB OUTPUT FILE

The job shown in figure 4-2 produces job output similar to figure 4-3.

```
------------------------------------
NQS 4.0.0 BATCH QUEUE SUMMARY: sn1101
------------------------------------
QUEUE NAME       LIM TOT ENA STS QUE RUN  WAI HLD ARR EXI  TAPES QUICKFL USR GRP
---------------- --- --- --- --- --- ---  --- --- --- ---  -- -- --- --- --- ---
b_10_1            15   0 yes  on   0   0    0   0   0   0  --  0  --   0  10  10
b_25_2            10   0 yes  on   0   0    0   0   0   0  --  0  --   0   2   2
b_50_5             5   0 yes  on   0   0    0   0   0   0  --  0  --   0   1   1
b_200_8            1   0 yes  on   0   0    0   0   0   0  --  0  --   0   1   1
b_1200_12          1   0 yes  on   0   0    0   0   0   0  --  0  --   0   1   1
b_15000_15         1   0 yes  on   0   0    0   0   0   0  --  0  --   0   1   1
st_express        20   0 yes  on   0   0    0   0   0   0  --  0  --   0   5   5
st_little          5   0 yes  on   0   0    0   0   0   0  --  0  --   0   2   2
st_medium          5   0 yes  on   0   0    0   0   0   0  --  0  --   0   2   2
st_large           2   0 yes  on   0   0    0   0   0   0  --  0  --   0   2   2
st_exlarge         2   1 yes  on   0   1    0   0   0   0  --  0  --   0   2   2
single_200_65      1   0 yes  on   0   0    0   0   0   0  --  0  --   0   1   1
qlfoo2245229       5   0  no  on   0   0    0   0   0   0  --  0  --   0  20  20
qlfoo23A_24856     1   0 yes off   0   0    0   0   0   0  --  0  --   0  --  --
qlfoo4A2_54397     5   0  no  on   0   0    0   0   0   0  --  0  --   0  20  20
qlfoo              5   0  no  on   0   0    0   0   0   0  --  0  --   0  20  20
qlfoo2430282       1   0 yes  on   0   0    0   0   0   0  --  0  --   0  --  --
qlfoo2832831       1   0 yes  on   0   0    0   0   0   0  --  0  --   0  --  --
qlfoo2833146       1   0 yes  on   0   0    0   0   0   0  --  0  --   0  --  --
---------------- --- --- --- --- --- ---  --- --- --- ---  -- -- --- --- --- ---
<GLOBAL>          20   1           0   1    0   0   0   0  --  0  --   0  --  --
---------------- --- --- --- --- --- ---  --- --- --- ---  -- -- --- --- --- ---
------------------------------------
NQS 4.0.0 PIPE QUEUE SUMMARY: sn1101
------------------------------------
QUEUE NAME       LIM TOT ENA STS QUE ROU  WAI HLD ARR DEP  DESTINATIONS
---------------- --- --- --- --- --- ---  --- --- --- ---  ------------
batch              1   0 yes  on   0   0    0   0   0   0
                                                           st_express@sn1101
                                                           st_little@sn1101
                                                           st_medium@sn1101
                                                           st_large@sn1101
                                                           st_exlarge@sn1101
                                                           b_10_1@sn1101
                                                           b_25_2@sn1101
                                                           b_50_5@sn1101
                                                           b_200_8@sn1101
                                                           b_1200_12@sn1101
                                                           b_15000_15@sn1101
snq1               1   0 yes  on   0   0    0   0   0   0  batch@snq1
sn2012             1   0 yes  on   0   0    0   0   0   0  batch@sn2012
sn218              1   0 yes  on   0   0    0   0   0   0  batch@sn218
sjs                1   0 yes  on   0   0    0   0   0   0  single_200_65@sn1101
---------------- --- --- --- --- --- ---  --- --- --- ---  -------------
<GLOBAL>          20   0           0   0    0   0   0   0
---------------- --- --- --- --- --- ---  --- --- --- ---  -------------
```

Figure 4-3. UNICOS Job Output

```
no device queues
for.f: ? for.f: File exists.
::::::::::::::
for.f
::::::::::::::
      program test
      j=50
      do 10 i=1,10
      j=j+1
      print 50,j
10    continue
      stop
50    format(' ',I3)
      end
 CFT77 - CFT77    VERSION 2.0      05/18/88 05:11:20
 CFT77 - COMPILE TIME     .073 SECONDS
 CFT77 - 9 SOURCE LINES
 CFT77 - 0 ERRORS, 0 OTHER MESSAGES
 CFT77 - CODE: 12 WORDS, DATA: 31 WORDS
  51
  52
  53
  54
  55
  56
  57
  58
  59
  60
   STOP              in TEST
logout
```

Figure 4-3. UNICOS Job Output (continued)

5. TRANSFERRING DATA BETWEEN UNIX AND COS

This section describes how to transfer (stage) data between the front-end system and COS. The information is presented as follows:

- Transferring data from UNIX to COS
  - Moving UNIX files to COS - save
  - Moving files with the ACQUIRE job control statement
  - Moving files with the FETCH job control statement
- Transferring data from COS to UNIX

5.1 TRANSFERRING DATA FROM UNIX TO COS

This subsection describes how to transfer data from the front-end computer system to COS using one of the following methods:

- As a UNIX user, with the save station command
- Through a COS job, with the COS ACQUIRE or FETCH job control statements

This section describes each of these facilities and provides examples of common usage. If you are a new user of COS, see appendix A for a discussion of permanent and temporary dataset characteristics.

All files sent to COS must have other (public) read access.

5.1.1 MOVING UNIX FILES TO COS - save

The save station command queues a request for the station to stage a file from the UNIX front-end computer system to COS. At the Cray system, the dataset is entered onto a Cray mainframe mass storage device. The dataset is saved with the owner value of SYSTEM and public access mode R:W:M (read, write, and maintenance permissions); you must specify OWN=SYSTEM to access the dataset after it is saved. (You can use the COS ACQUIRE or FETCH job control statements instead of the save command if permanent dataset privacy is enabled on the Cray system.) You can enter the save station command from console mode with the following format.

```
 ________________________________________
|                                        |
| save  pathname|-  pdn  [id|- [ed]]     |
|________________________________________|
```

pathname	Path name of the file to be staged to COS. The *pathname* can begin with a slash (/) to specify a full UNIX path name or a tilde (~) to specify a path beginning with a specified user's home directory. If *pathname* does not begin with a slash or tilde (/ or ~), the path starts in the current directory. If *pathname* is a dash (-), the save station command reads the file from **stdin** (standard input).
pdn	Permanent dataset name of the COS job file to be saved; 1 to 15 characters (A through Z, 0 through 9, $, % or @). The first character must be alphabetic. If *pdn* is an invalid COS dataset name, and the dataset is accessed after it is saved, the *pdn* specified in the ACCESS statement must be entered exactly as it is specified on the save command line and must be enclosed in single quotes.
id	User ID of the permanent dataset to be saved; 1 to 8 alphanumeric characters. Specify a dash to omit this field.
ed	Edition number of the dataset; from 1 to 4095. The default is 1. If a dataset with the specified edition number already exists, the dataset transfer is canceled. When specifying *ed*, you must also include either a user ID or hyphen (-) in the *id* field.

Examples:

In the following example, the **cft** file from the current directory is saved with a permanent dataset name of **job1** and an edition number of 6. The dataset is then accessed with the ACCESS job control statement within the **cft** file.

```
    $ cs save cft job1 - 6

    ACCESS,DN=cft,PDN='job1',OWN=SYSTEM,ED=6.
```

In the following example, the **/u/source/c** file is saved with a permanent dataset name of **compile** and an ID of U5066. The dataset is then accessed with the ACCESS job control statement within the **/u/source/c** file.

```
    $ cs save /u/source/c compile U5066

    ACCESS,DN=c,PDN='compile',OWN=SYSTEM,ED=1,ID=U5066.
```

5.1.2 MOVING FILES WITH THE ACQUIRE JOB CONTROL STATEMENT

The COS ACQUIRE job control statement allows you to make a dataset permanent and local to a job. FETCH also allows you to make the output of a shell command local to a COS job. In order to do either of these actions, you must include an ACQUIRE job control statement in your job running on the Cray system.

When you issue an ACQUIRE statement, COS determines whether the requested dataset is permanently resident on Cray mass storage or resident on the front-end computer system. If COS determines that the requested dataset is already permanently resident on Cray mass storage, the dataset is made local to the COS job. If the requested dataset is not on Cray mass storage, COS sends the request for the dataset to the front-end system. COS then stages the dataset to Cray mass storage and makes the dataset permanent and local to the job making the request.

```
ACQUIRE,DN=dn[,AC=ac][,ID=uid][,MF=mf][,DF=df],TEXT='text'.
```

DN=*dn*	Local dataset name
AC=*ac*	Acquisition code. The source from which the dataset is to be acquired. If the AC parameter is omitted, the default is ST.

The 2-character alphanumeric code *ac* describes the source for the dataset as follows:

IN	Input (job) dataset. Use the SUBMIT control statement to run the job.
IT	Intertask communication
MT	Magnetic tape at the front-end system designated by the MF parameter
ST	Staged dataset from the front-end system designated by the MF parameter

NOTE

The dataset acquisition codes previously noted are only a convention. Actual dataset acquisition is determined by the front-end system.

ID=*uid*	User identification
MF=*mf*	Mainframe identifier for the front-end computer system; 2 alphanumeric characters. The default is the front end of job origin. This must be a batch station ID.
DF=*df*	Dataset format:

CB	Character blocked; the dataset is sent from the front-end system in COS blocked format. CB is the default.
TR	Transparent; the data is treated as a continuous byte string.
BB	Binary blocked. Use this option when working with unformatted sequential Fortran files. Each record has a 4-byte header and trailer indicating the length of the record in bytes.

TEXT='*text*'

Text to be passed to the front-end system. *text* is the UNIX path name of the file to be acquired. The path name should be the full UNIX path name (it must begin with either a / or the home directory character ~). *text* must be enclosed in single quotes.

If the *text* field begins with an exclamation point (!), the specified text points to a shell command that will be executed on the front-end system. Standard output for that command is assigned to the ACQUIRE statement.

The specified shell command runs as the station operator in the station root directory. Shell files should be specified as full path names and must have public read permission. The environment used to execute the command is the default environment for the station operator; therefore, any changes to that environment must be explicitly defined in the shell command specified in the *text* field. For example, if your shell command expects to find the **localpr** file in **/usr/local/bin**, and **/usr/local/bin** is not defined in the station operator's default PATH, you must either redefine PATH within the shell script or reference **localpr** as **/usr/local/bin/localpr**.

Examples:

By taking advantage of system defaults, the following example accesses a file from the front end system using only the parameters for dataset name (DN) and text (TEXT). The name of the permanent dataset on the Cray system will be MYFILE.

```
ACQUIRE,DN=MYFILE,TEXT='/u/abc/source'.
```

In the following example, the COS dataset is staged in transparent mode:

```
ACQUIRE,DN=MYFILE,DF=TR,TEXT='/u/abc/source'.
```

The following example specifies the permanent dataset identifier as P5906 and the front-end identifier as UX:

```
ACQUIRE,DN=MYFILE,ID=P5906,MF=UX,DF=TR,TEXT='/u/abc/source'.
```

5.1.3 MOVING FILES WITH THE FETCH JOB CONTROL STATEMENT

The COS FETCH job control statement allows you to make a file that resides on a front-end computer system local to a COS job. FETCH also allows you to make the output of a shell command local to a COS job. In order to use a FETCH job control statement, you must include one in your job running on the Cray system. Unlike ACQUIRE, FETCH does not make a dataset permanent on Cray mass storage.

The interactive station command ~f causes a FETCH job control statement to be sent to COS. For more information on ~f, see subsection 7.6, Sample Interactive Terminal Session, or the UNIX Station Command Reference Manual.

```
FETCH,DN=dn[,AC=ac][,MF=mf][,DF=df],TEXT='text'.
```

DN=*dn*	Local dataset name
AC=*ac*	Acquisition code. The source from which the dataset is to be acquired. If the AC parameter is omitted, the default is ST. The 2-character alphanumeric code ac describes the source for the dataset as follows: IN Input (job) dataset. Use the SUBMIT control statement to run the job.

IT	Intertask communication
MT	Magnetic tape at the front-end system designated by the MF parameter
ST	Staged dataset from the front-end system designated by the MF parameter

NOTE

The dataset acquisitions previously noted are by convention only. Actual dataset acquisition is determined by the front-end system.

MF=*mf* Mainframe identifier for the front-end computer system. The default is the front end of job origin. This must be a batch station ID.

DF=*df* Dataset format:

CB	Character-blocked; the dataset is sent from the front-end system in COS blocked format. CB is the default.
TR	Transparent; the data is treated as a continuous byte string.
BB	Binary-blocked. Use this option when working with unformatted sequential Fortran files. Each record has a 4-byte header and trailer indicating the length of the record in bytes.

TEXT='*text*'

Text to be passed to the front-end system. *text* is a UNIX path name. It must begin with a / or the home directory character ~. *text* must be enclosed in single quotes.

If the *text* field begins with an exclamation point (!), the specified text points to a shell command which will be executed on the front-end system. Standard output for that command is assigned to the FETCH statement.

The specified shell command runs as the station operator in the station root directory. Shell files should be specified by full path names and must have public read and execute permission. The environment used to execute the command is the default environment for the station operator; therefore, any changes to that environment must be explicitly defined in the shell command specified in the *text* field. For example, if your shell command expects to find the **localpr** file in **/usr/local/bin**, and **/usr/local/bin** is not defined in the station operator's default PATH, you must either redefine PATH within the shell script or reference **localpr** as **/usr/local/bin/localpr**.

Examples:

The following example accesses a file from the front end system using only the parameters for dataset name (DN) and text (TEXT). The name of the local dataset is APPLE, and the name of the UNIX file is **/u/abc/source**:

```
FETCH,DN=APPLE,TEXT='/u/abc/source'.
```

In the following example, the COS dataset is staged in transparent mode:

```
FETCH,DN=APPLE,DF=TR,TEXT='/u/abc/source'.
```

The following example specifies the front-end identifier as UX:

```
FETCH,DN=APPLE,MF=UX,DF=TR,TEXT='/u/abc/source'.
```

In the following example, the same COS dataset is staged from the **/u/abc** home directory, using the home directory character ~:

```
FETCH,DN=APPLE,DF=TR,TEXT='~abc/source'.
```

In the following example, the content of a tape is staged to COS:

```
FETCH,DN=APPLE,TEXT='!dd if=/dev/rmt/0m'.
```

The following example shows how to execute a shell command in the TEXT field. The FETCH job control statement uses the standard output of the shell command as the contents of the file to be transferred:

```
FETCH,DN=APPLE,MF=SX,TEXT='!ls -alt'.
```

5.2 TRANSFERRING DATA FROM COS TO UNIX

The COS DISPOSE job control statement allows you to transfer (stage) a local dataset from COS to the front-end computer system. DISPOSE also allows you to stage a local dataset from COS and use it as input to a shell command on the front-end system. In order to use a DISPOSE job control statement, you must include one in your job running on the Cray system.

Datasets staged from COS to the front-end system are placed in your user default output directory (which has other (public) write access), unless you specify otherwise with the TEXT parameter on the DISPOSE control statement. If you do not have a user default output directory, all files received from COS are placed in a station default output directory (specified by your system administrator). See the description of the TEXT parameter in this subsection for more information.

When you are in COS interactive mode, the **-d** interactive station command sends a DISPOSE control statement to COS. For more information on **-d**, see the UNIX Station Command Reference Manual.

```
DISPOSE,DN=dn[DC=dc][,MF=mf][,DF=df][,SF=sf][,TEXT='text']

   [,TID='tid'][,WAIT][,NRLS].
```

DN=*dn*	Local dataset name
DC=*dc*	Disposition code. Disposition to be made of the dataset. If the DC parameter is omitted, the default is PR (print).

The 2-character alphanumeric code *dc* describes the destination of the dataset as follows:

IN	Input (job) dataset. Dataset is queued as a job on the mainframe specified with the MF parameter.
IT	Intertask communication
MT	Dataset is to be written on magnetic tape at the front-end system designated by the MF parameter
PR	Print dataset. Dataset is printed on a printer available at the front-end system designated by the MF parameter.

PT Plot dataset. Dataset is plotted on any available plotter at the front-end system designated by the MF parameter.

PU Punch dataset. Dataset is punched on any card punch available at the front-end system designated by the MF parameter.

SC Scratch dataset. Dataset is released, unless another DISPOSE request is still pending on the dataset. This parameter has the same effect as RELEASE,DN=*dn*.

ST Staged to the front-end system. Dataset is saved at the front-end system designated by the MF parameter.

NOTE

The dataset dispositions previously noted are by convention only. With the exception of SC, actual dataset disposition is determined by the destination front-end system.

MF=*mf* Mainframe identifier for the front-end computer system; 2 alphanumeric characters. The default is the front end of job origin.

DF=*df* Dataset format:

CB Character-blocked; the dataset is sent to the front-end system in COS blocked format. CB is the default.

TR Transparent; the data is treated as a continuous byte string.

BB Binary-blocked. Use this format when working with unformatted sequential Fortran files. Each record has a 4-byte header and trailer indicating the length of the record in bytes.

SF=*sf* Special form information to be passed to the front-end system, as follows:

REPLACE	Specifies replacement of an existing file on the front-end system
APPEND	Specifies that the dataset will be appended to an existing file on the front-end system

The default is to abort the transfer if the named file already exists.

If SF is not specified, and the file does not already exist on the front-end system, a new file is created.

TEXT='*text*'

Specifies a UNIX path name to which data from the Cray system is disposed.

If the *text* field begins with an exclamation point (!), the specified text points to a shell command that will be executed on the front-end system. Standard input for that command is assigned to the DISPOSE statement.

The specified shell command runs as the station operator in the station root directory. Shell files should be specified by full path names and must have public read and execute permission. The environment used to execute the command is the default environment for the station operator; therefore, any changes to that environment must be explicitly defined in the shell command specified in the *text* field. For example, if your shell command expects to find the **localpr** file in **/usr/local/bin**, and **/usr/local/bin** is not defined in the station operator's default PATH, you must either redefine PATH within the shell script or reference **localpr** as **/usr/local/bin/localpr**.

To specify a full UNIX path name, *text* must begin with a / or the home directory character ~. If the file cannot be placed in the specified location, the file is placed in the station default output directory (usually ST_LOSTDIR).

If you do not specify TEXT, and the job was submitted through the station, the file is returned to your default output directory (defined in $CRAYOUT). If $CRAYOUT was not defined at the time of job submission, the data file is returned to a subdirectory in your home directory ($HOME) named ST_DEFSUFFIX/*filename*. If the station cannot return the file to either the default output directory or the $HOME/ST_DEFSUFFIX directory, the file is placed in the station default output directory (ST_LOSTDIR).

If you do not specify TEXT, and the job was not submitted through the UNIX station, the station interprets the terminal identifier (TID) as a UNIX front-end user name. If the station cannot resolve the TID as a valid user name or write to $HOME/ST_DEFSUFFIX, the file is placed in the station lost and found directory.

TID=*tid*	Default terminal identifier (TID). The default is the login name of the user submitting the job.
WAIT	Specifies that the job does not resume processing until the disposed dataset has been staged to the front-end system. If the front-end system cancels the transfer, the waiting job is aborted, and job step abort processing occurs as described in the COS Reference Manual. If there is a problem with the transfer, the station places a message in the job's log file if the job is still running. The default is a site-dependent option.
NRLS	No release option. The dataset remains local to the job after the DISPOSE request has been processed. When NRLS is specified, the dataset cannot be written to until the transfer to the specified front end is completed. Therefore, WAIT should be used with NRLS. The default is to release a dataset after queueing it for transfer.

Examples:

The following example stages a local dataset from COS to the front-end system using only the parameter for the dataset name (DN). The name of the local dataset is UFILE.

```
DISPOSE,DN=UFILE.
```

In the following example, the COS dataset is staged in transparent mode and will be placed in the UNIX file /usr/abc/source:

```
DISPOSE,DN=UFILE,DF=TR,TEXT='/usr/abc/source'.
```

The following example appends the COS dataset to an existing file on the front-end system:

```
DISPOSE,DN=UFILE,DF=TR,SF=APPEND,TEXT='/usr/abc/source'.
```

The following example stages a COS dataset to the front-end system as standard input to the **lpr** command, which sends the dataset to the printer:

```
    DISPOSE,DN=UFILE,DF=TR,TEXT='!lpr'.
```

The following example stages the same COS dataset to the **/usr/abc** home directory, using the home directory character ~:

```
    DISPOSE,DN=UFILE,DF=TR,TEXT='~abc/source'.
```

The following example uses the staged file as standard input to the shell command **cat**:

```
    DISPOSE,DN=UFILE,MF=SX,TEXT='!cat >> /u/abc/file2'.
```

6. TRANSFERRING DATA BETWEEN UNIX AND UNICOS

This section describes how to transfer files between the UNIX front-end system and UNICOS. All UNICOS commands in this section are described in the UNICOS User Commands Reference Manual.

This section uses the following UNICOS conventions:

- Each UNICOS data transfer command appears in lowercase, boldface type and is followed by a numeric designation identifying the UNICOS manual in which it appears. For example, the command description for **acquire**(1) appears in the UNICOS User Commands Reference Manual. (When using UNICOS interactively, you can access the on-line man pages for all UNICOS commands with the **man**(1) command.)

- Hyphens separate command options.

- When the complete syntax of the data transfer statement is shown, **boldface** strings are literals and are to be typed in just as they appear; *italic* strings represent user-supplied information. Brackets [] around an option or an operand indicate that it is optional. See the UNICOS User Commands Reference Manual for a complete description of the UNICOS command format.

Case is significant in UNICOS. Enter all commands in lowercase, whether they are submitted in a batch file or entered interactively.

This section describes the following UNICOS data transfer commands:

- **acquire** - Transfers a file from a front-end system to UNICOS

- **dispose** - Transfers a file from UNICOS to the front-end system

- **fetch** - Transfers a file from a front-end system to UNICOS; if the file already exists, the system overwrites it.

- **qsub** - Submits a batch request to the Network Queueing System (NQS)

- **uscpqsub** - Submits a job to NQS from a job that has been spawned by the UNICOS Station Call Processor (USCP)

NAME

acquire - Requests a file from the front-end system; checks for the file's existence on UNICOS

SYNOPSIS

acquire *localpath* [**-n** *sfn*] [**-i** *termid*] [**-m** *mf*] [**-d** *ac*] [**-f** *fm*]
[**-t** '*text*'] [**-u** *user*]

DESCRIPTION

The **acquire** command requests a file from a front-end station. It differs from **fetch**(1) in that it first checks for the existence of the specified file on UNICOS. If the file exists at the time you invoke the **acquire** command, the command returns directly with a zero status. If the file does not exist, **acquire** requests that the USCP retrieve it.

USCP uses station protocol to request a file from the designated station (specified by the -m option). The **acquire** command then waits until the transfer status has been determined to complete the transfer. If USCP receives the file, the exit status is 0. If not, the transfer is aborted; **stderr** should indicate the reason. If the station returns a postpone status, the **acquire** command resets the request for USCP and waits for the transfer status.

The acquire command accepts the following arguments.

localpath A path name (either full or relative to the current directory) indicating the location at which the requested file is to reside when the transfer is complete. The requesting user must have permission to write to *localpath*.
This operand is required.

-n *sfn* The 1-15 character name associated with the requested file on the specified front-end system. This argument is stored in the request record PDN field. If you do not specify *sfn*, the field is filled with the file name from *localpath*.

-i *termid* The 1-8 character terminal ID associated with the requested file on the specified front-end system. If you do not specify *termid*, the default is the terminal ID associated with the requesting user on the front-end station from which the request originated.

DESCRIPTION (continued)

-m *mf*	A 2-character front-end ID for a station that has access to the requested file. If you do not specify the mainframe, the stored ID of the station from which the request originated is used.
-d *ac*	A 2-character acquisition code interpreted by the front-end system. Valid codes are as follows:

IN	File is to be executed as a job by the front-end system
ST	File is to be saved by the front-end system

-f *fm*	A 2-character file format code. Valid codes are as follows:

CB	Character-blocked. USCP deblocks the station-blocked file to an ASCII file; record control words are converted to new lines.
BB	Binary-blocked. USCP deblocks the station-blocked file to a binary file.
TR	Transparent. The file is saved without conversion.
TB	Transparent binary-blocked. The file is saved without conversion; use this format to move the blocked file without deblocking.
TC	Transparent character-blocked. The file is saved without conversion; use this format to move the character-blocked file without deblocking to the Cray system.
UD	UNICOS data; the default. The station performs character conversion to ASCII, if necessary, and sends file to the Cray system.

-t '*text*'	Text to be interpreted by the station for processing the request. The field contains a UNIX path name of a file to be acquired; it can be a full or relative path name. The *text* field should be enclosed in single quotation marks (') to prevent it from being interpreted by the shell. If you do not specify *text*, the command is accepted, but the station logs an error message, and the transfer fails.

DESCRIPTION (continued)

If the *text* field begins with an exclamation point (!), the specified text points to a shell command that will be executed on the front-end system. Standard output for the shell command is assigned to the **acquire** command.

The specified shell command runs as the station operator in the station root directory. Files should be specified by full path names and must have public write permission. Because the environment used to execute the command is the default environment for the station operator, any changes to that environment must be explicitly defined in the shell command. For example, if your shell command expects to find the **localpr** file in **/usr/local/bin**, and **/usr/local/bin** is not defined in the station operator's default PATH, you must either redefine PATH within the shell script or reference **localpr** as **/usr/local/bin/localpr**.

-u *user* The user ID associated with the requested file on the specified front-end system. If you do not specify *user*, this field is left blank for the request.

NOTES

If you are not accessing the Cray system through USCP, the defaults for *termid* and *mf* do not exist. Therefore, the request is queued whether or not the specified mainframe ID belongs to a currently active station. If the associated station is not active or has no streams assigned (that is, an interactive-only station), the user process waits indefinitely.

SEE ALSO

fetch(1), **dispose**(1), **uscpqsub**(1), **uscproute**(1)
UNICOS Primer, publication SG-2010
UNICOS Station Call Processor (USCP) Administrator's Guide, publication SG-2072

NAME

dispose - Disposes a file to a front-end station from a Cray system

SYNOPSIS

dispose *localpath* [**-n** *sfn*] [**-i** *termid*] [**-m** *mf*] [**-d** *dc*] [**-f** *fm*] [**-t** '*text*'] [**-s** *special*] [**-u** *user*]

DESCRIPTION

The **dispose** command creates a request file for the USCP. If slot information is associated with the requesting user, it is also copied into the request file. USCP uses station protocol to negotiate the file transfer from the Cray system to the designated station (specified by the **-m** option). The **dispose** command then waits for the transfer to complete.

localpath	The path name (either full or relative to the current directory) of the file to be disposed. This must be a path name from which the requesting user has permission to read.
-n *sfn*	The name to be associated with the file when it is received by the front end. Only 15 characters are significant. If you do not specify *sfn*, the field is filled with the file name from *localpath*.
-i *termid*	The terminal ID to be associated with the file on the specified front-end system. If you do not specify *termid*, the default is the stored terminal ID associated with the requesting user on the front-end station from which the request originated.
-m *mf*	A 2-character front-end ID for the station to which the file is transferred. If you do not specify the mainframe, the stored ID of the station from which the request originated is used.
-d *dc*	A 2-character disposition code. Valid codes are as follows:

ST	File is staged to a front-end system.
MT	File is disposed to a magnetic tape.
PR	File is disposed to a printer.

DESCRIPTION (continued)

PU	File is disposed to a card punch.
PT	File is disposed to a plotter.
IT	File is flagged as intertask data and handled by the receiving station.

-f *fm* A 2-character file format code. Valid formats are as follows:

CB	Character-blocked. USCP blocks the file for delivery to the front-end station. New-line characters are used to denote the end of the record.
BB	Binary-blocked. USCP blocks the file for delivery to the front-end station. Characters are not used to determine record size. The default is 510 words.
TR	Transparent. Data is sent without conversion.
TB	Transparent binary-blocked. Data is sent without conversion; the front-end station is informed that the format is BB.
TC	Transparent character-blocked. Data is sent without conversion; the front-end station is informed that the format is CB.
UD	UNICOS data; the default. Data is sent without conversion; the front-end station does character conversion if necessary.

-t '*text*' Text to be interpreted by the specified station for **dispose** processing. The field contains a UNIX path name that identifies the file to which data from the Cray system is disposed. The text should be enclosed in single quotation marks (') to prevent it from being interpreted by the shell. If you do not specify this option, the *text* field is zero-filled.

DESCRIPTION (continued)

If *text* begins with an exclamation point (!), the specified text points to a shell command to be executed on the front-end system. Standard input for that command is assigned to the **dispose** command.

The specified shell command runs as the station operator in the station root directory. Shell files should be specified by full path names and must have public read and execute permission. The environment used to execute the command is the default environment for the station operator; therefore, any changes to that environment must be explicitly defined in the shell command specified in the *text* field. For example, if your shell command expects to find the **localpr** file in **/usr/local/bin**, and **/usr/local/bin** is not defined in the station operator's default PATH, you must redefine PATH within the shell script or reference **localpr** as **/usr/local/bin/localpr**.

To specify a full UNIX path name, *text* should begin with a / or the home directory character ~. If the file cannot be placed in the specified location, it is placed in the station default output directory (usually ST_LOSTDIR).

If you do not specify TEXT, and the job was submitted through the station, the file is returned to your default output directory (defined in $CRAYOUT). If $CRAYOUT was not defined at the time of job submission, or if the station cannot write to the specified path, the data file is returned to a subdirectory of your home directory ($HOME/ST_DEFSUFFIX, usually $HOME/tmp). If the station cannot return the file to either the default output directory or the home directory, the file is placed in the station default output directory.

If you do not specify TEXT, and the job was not submitted through the UNIX station, the station interprets the terminal identifier (TID) as a UNIX front-end user name. If the station cannot resolve the TID as a valid user name or write to $HOME/tmp, the file is placed in the station lost and found directory.

-s *special* The station-defined special forms option.

replace	Specifies replacement of an existing file on the front-end system

DESCRIPTION (continued)

append Specifies that the file will be appended to an existing file on the front-end system

If you do not specify this option, the special forms field is zero-filled. The default aborts the transfer if the named file already exists. If -s is not specified, and the file does not already exist on the front-end system, a new file is created.

-u *user* The user ID associated with the requested file on the specified front-end system. If you do not specify *user*, the field remains blank.

NOTES

If you are not accessing the Cray system through USCP, the defaults for *termid* and *mf* do not exist. Therefore the request is queued whether or not the mainframe ID you specified belongs to a currently active station. If the associated station is not active or does not have streams assigned (that is, an interactive-only station), then the user process waits indefinitely.

The C shell and the Bourne shell interpret the new-line character differently. The C shell ignores the new-line character. The Bourne shell concatenates the lines with the new-line character embedded. This can cause problems when you enter the *text* field, because *text* must be a continuous string.

If you abort a **dispose** command after the file transfer has begun, the transfer runs to completion.

EXAMPLES

```
# dispose file to UNIX machine TB
dispose outfile -mTB -t/u/tom/nqs/outfile
```

SEE ALSO

acquire(1), **fetch**(1), **uscpqsub**(1), **uscproute**(1)
UNICOS Primer, publication SG-2010
UNICOS Station Call Processor (USCP) Administrator's Guide, publication SG-2072

NAME

fetch - Requests a file from a front-end station

SYNOPSIS

fetch *localpath* [**-n** *sfn*] [**-i** *termid*] [**-m** *mf*] [**-d** *ac*] [**-f** *fm*]
[**-t** '*text*'] [**-u** *user*]

DESCRIPTION

The **fetch** command creates a request file for USCP. An existing file of the given name on UNICOS is overwritten. The slot information associated with the requesting user is also copied into the request file.

USCP uses station protocol to request a file from the designated station (specified by the -m option). The **fetch** process then waits until the transfer status has been determined to complete the transfer. The transfer status is returned when a negative reply is received from the station (requested file did not transfer) or when a positive reply is received from the station (requested file was saved on the Cray system). If the station returns a postpone status, the **fetch** process resets the request for USCP and waits for the transfer status.

The **fetch** command accepts the following arguments.

localpath A path name (either full or relative to the current directory) indicating the location at which the requested file is to reside when the transfer is complete. The user must have permission to write to *localpath*. This is a required operand.

-n *sfn* A 1-15 character name associated with the requested file on the specified front-end system. This argument is stored in the request record PDN field. If you do not specify *sfn*, this field is filled with the file name from *localpath*.

-i *termid* A 1-8 character terminal ID associated with the requested file on the specified front-end system. If you do not specify *termid*, the default is the stored terminal ID associated with the requesting user on the front-end station from which the request originated.

-m *mf* A 2-character front-end ID for a station that has access to the requested file. If you do not specify the mainframe, the stored ID of the station from which the request originated is used.

DESCRIPTION (continued)

-d *ac* A 2-character acquisition code interpreted by the front-end system. Valid codes are as follows:

- **MT** File is on magnetic tape.
- **IN** File is to be executed as a job.
- **ST** File is to be saved.

-f *fm* A 2-character file format code. Valid formats are as follows:

- **CB** Character-blocked. USCP deblocks a station-blocked file to an ASCII file; record control words are converted to new lines.
- **BB** Binary-blocked. USCP deblocks a station-blocked file to a binary file.
- **TR** Transparent. The file is saved without conversion.
- **TB** Transparent binary-blocked. The file is saved without conversion; use this format to move the blocked file without deblocking.
- **TC** Transparent character-blocked. The file is saved without conversion; use this format to move the character-blocked file without deblocking to the Cray system.
- **UD** UNICOS data; the default. The station converts the characters to ASCII, if necessary, and sends the file to the Cray system.

-t '*text*' Text to be interpreted by the specified station for processing of the request. The field can contain a full UNIX path name of the file to be fetched. It should begin with a / or with the home directory character ~. The *text* should be enclosed in single quotation marks (') to prevent it from being interpreted by the shell. If you do not specify *text*, the command is accepted, but the station logs an error message and the transfer fails.

DESCRIPTION (continued)

If the *text* field begins with an exclamation point (!), the specified text points to a shell command that will be executed on the front-end system. Standard output for that command is assigned to the **fetch** command.

The specified shell command runs as the station operator in the station root directory. Shell files should be specified by full path names and must have station/operator read and execute permission. The environment used to execute the command is the default environment for the station operator; therefore, any changes to that environment must be explicitly defined in the shell command specified in the *text* field. For example, if your shell command expects to find the **localpr** file in **/usr/local/bin**, and **/usr/local/bin** is not defined in the station operator's default PATH, you must either redefine PATH within the shell script or reference **localpr** as **/usr/local/bin/localpr**.

-u *user* The user ID associated with the requested file on the specified front-end system. If you do not specify *user*, this field remains blank.

NOTES

If you are not accessing the Cray system through USCP, there are no defaults for *termid* and *mf*. Therefore the request is queued whether or not the mainframe ID you specified belongs to a currently active station. If the associated station is not active or has no streams assigned (that is, an interactive-only station), the user process waits indefinitely.

The C shell and the Bourne shell interpret the new-line character differently. The C shell ignores the new-line character. The Bourne shell concatenates the lines with the new-line character embedded. This can cause problems when you enter the *text* field, because *text* must be a continuous string.

If you abort a **fetch** command after the file transfer has begun, the transfer continues to run to completion.

SEE ALSO

acquire(1), **dispose**(1), **uscpqsub**(1), **uscproute**(1)
UNICOS Primer, publication SG-2010
UNICOS Station Call Processor (USCP) Administrator's Guide, publication SG-2072

NAME

qsub - Submits a batch request to the Network Queueing System

SYNOPSIS

qsub [**-a** *date-time*] [**-e** *file*] [**-eo**] [**-ke**] [**-ko**] [**-lf** *limit*]
[**-lF** *limit*] [**-lm** *limit*] [**-lM** *limit*] [**-ln** *limit*] [**-lQ** *limit*]
[**-lt** *limit*] [**-lT** *limit*] [**-lU** *limit*] [**-mb**] [**-me**] [**-mu** *user-name*]
[**-nc**] [**-nr**] [**-o** *file*] [**-p** *priority*] [**-q** *queue-name*] [**-r** *request-name*]
[**-re**] [**-ro**] [**-s** *shell-name*] [**-x**] [**-z**] [*file*]

DESCRIPTION

The **qsub** command submits a file, containing a shell script, as a batch request to the Network Queueing System (NQS).

The **qsub** command accepts the following arguments:

-a	Runs the request after the stated time
-e	Directs standard error output to the specified destination
-eo	Directs standard error output to the standard output destination
-ke	Keeps standard error output on the execution machine
-ko	Keeps standard output on the execution machine
-lf	Establishes per-process file size limits
-lF	Establishes per-request permanent file size limits
-lm	Establishes per-process memory size limits
-lM	Establishes per-request memory size limits
-ln	Establishes per-process nice execution value limits
-lQ	Establishes per-request secondary data segment limits
-lt	Establishes per-process CPU time limits
-lT	Establishes per-request CPU time limits
-lU	Establishes per-request CPU tape drive limit
-mb	Sends mail when the request begins execution
-me	Sends mail when the request ends execution
-mu	Sends mail for the request to the stated user
-nc	Specifies that the batch request is irrecoverable
-nr	Specifies that the batch request is not rerunnable
-o	Directs standard output to the stated destination
-p	Specifies intraqueue request priority
-q	Queues requests in the stated queue
-r	Assigns the stated request name to the request
-re	Remotely accesses the standard error output file
-ro	Remotely accesses the standard output file
-s	Specifies the shell to interpret the batch shell script
-x	Exports all environment variables with the request
-z	Submits the request silently

DESCRIPTION (continued)

During batch request submission, the script file *file* is spooled so that subsequent changes do not affect the queued batch request.

Upon successful completion of the batch request submission, NQS responds with the request-identifier associated with the batch request and the name of the NQS queue that initially accepted the request. The request-identifier, which identifies your batch request uniquely throughout the NQS network, has the format *sequence-number.host*.

NQS assigns a *sequence number* to your batch request; *host* is the name of the host that submitted the request.

If the **-z** option is specified, the response from **qsub** is suppressed.

By default, NQS returns the output produced by your request (stdout) and any error messages (stderr) to the machine and directory from which you submitted the request. The **stderr** and **stdout** files can be redirected by use of the appropriate **qsub** options.

Options can be embedded within *file* before the first executable command of the shell script if they are preceded with the **# QSUB** string. The characters **QSUB** must not be separated by blank spaces.

The following example specifies that the batch request should be submitted no earlier than 11 P.M. on Tuesday:

```
# QSUB-a 11pm tues.
```

For compatibility with earlier systems, the **# @$** string is acceptable as an alternative to **# QSUB**. If the same option appears in the batch file and on the command line, the command-line option takes precedence.

You can include comments in the script file by preceding them with a # character. You can append comments to an option by preceding them with the # character as follows:

```
# QSUB-a 11pm tues. # Sends job at 11 p.m. on Tuesday
```

DESCRIPTION (continued)

The following is a typical extended example of the use of embedded options within the script file:

```
     # Batch request shell script example:
     #
     # QSUB-q batch1          # Queue request to queue: batch1 by default
     # QSUG-z                 # Submits the request silently
     # QSUB                   # No more embedded options
     #
```

The **qsub** options can be specified in a batch request submitted from a front-end system through the UNIX station. The first line of *file* must specify the user name and password of the submitter, as required by the station. NQS options may be specified on successive lines in the format defined above.

EXAMPLES

The following is an example of a typical batch request file submitted to UNICOS from a front-end system through the UNIX station:

```
     # user=bill   pw=mypasswd
     # QSUB-r fortjob
     # QSUB-eo
     # QSUB-a 11 pm tues
     fetch for.f -nfort
     cft77 -eL for.f
     segldr for.o
     a.out
     dispose for.1 -nfort -t'ft=listing'
```

This job has a request name of **fortjob**. Its **stderr** and **stdout** files will be merged, and the request will be held until 11 P.M. on Tuesday. The job fetches a Fortran program from the station through which the job was submitted to UNICOS, executes the program, and disposes the results back to the station.

SEE ALSO

csh(1), **mail**(1), **qdel**(1), **qdev**(1), **qlimit**(1), **qmsg**(1), **qpr**(1), **qstat**(1), **sh**(1)
UNICOS Primer, publication SG-2010

NAME

uscpqsub - Submits a job to NQS from a job spawned by USCP

SYNOPSIS

uscpqsub [-m *mf*] [-i *tid*] *file*

DESCRIPTION

The **uscpqsub** command lets you submit batch jobs (specified by *file*) to NQS and specify the front-end system to which you want the output returned. Your job must meet the following requirements:

- You submitted the job to NQS through USCP.
- You are logged on interactively to UNICOS through USCP.

Jobs submitted with **uscpqsub** do not require a user ID record.

The **uscpqsub** command accepts the following arguments:

-m *mf*	Specifies a 2-character alternative station ID. The output is directed to the specified station. The default is the station of origin.
-i *tid*	Specifies a 1-8 character alternative *tid* to be used by the front-end station. The default is the original *tid*.
file	An NQS shell script containing the executable commands

RETURN VALUE

This command returns a 0 if successful and a nonzero value if unsuccessful.

SEE ALSO

qsub(1)
UNICOS Primer, publication SG-2010
Front-end Protocol Internal Reference Manual, publication SM-0042
UNICOS Station Call Processor (USCP) Administrator's Guide, publication SG-2072

7. USING THE COS INTERACTIVE FACILITY

This section provides the basic information you need to use the COS interactive facility. The section also includes a sample terminal session to provide you with on-line experience in using COS interactively.

This section discusses the following topics:

- The capabilities of the interactive facility
- Starting a COS interactive session
- Using COS job control language (JCL) and interactive station commands
- Ending a COS interactive session
- Getting help with the interactive station

See section 2, Getting Started, for information on setting up your terminal environment. Section 3, Submitting Jobs to COS, describes the COS batch processing facilities. Section 8, Using UNICOS Interactively, describes the interactive facility of UNICOS.

7.1 INTERACTIVE FACILITY OVERVIEW

The interactive facility of the UNIX station allows you to log on interactively to the Cray system to start a COS job. In the interactive environment, you can enter interactive station commands and COS job control or data statements one at a time after the default interactive prompt (!) from COS. You can observe the results of each COS job control statement before you proceed to the next, or redirect output to a file on the UNIX front-end computer system. For more information on COS JCL, see appendix A, Introduction to COS JCL, or the COS Reference Manual.

During program or utility execution, references to the COS datasets $IN and $OUT cause I/O at your terminal, unless interactive input or output has been redirected.

Interactive station commands allow you to perform functions such as transferring datasets and files and redirecting interactive input and output. See the UNIX Station Command Reference Manual for a description of the interactive station commands.

7.2 STARTING A COS INTERACTIVE SESSION

You can start an interactive session by entering the ias command from the shell. When you enter this command, the COS job name defaults to your front-end login name.

```
ias [-l] [-y] [-f char] [-n name] [-b id] [-s statid] [-i [file]]

  [-d lev] [-c crayid]
```

-l	Literal mode; null lines are sent to COS, and trailing blanks are saved on output from COS. By default, each null line is expanded to a single blank on input, and trailing blanks are stripped from output lines.
-y	Synchronous mode; the interactive station waits for a prompt from COS before sending a line of text. By default, input text is sent immediately and queued to COS. Synchronous mode is entered automatically when input is redirected to avoid overrunning COS buffer space.
-f *char*	Character to use for specifying commands local to ias. The default is a tilde (~).
-n *name*	COS job name; 1 to 7 characters in length. If *name* is not specified, the default is your front-end login name. *name* is converted to uppercase.
-b *id*	Batch station ID; a 2-character front-end ID of a batch station that will process FETCH and DISPOSE statements. This option is applicable only if the batch station is different from the station being used.
-s *statid*	A 2-character station mainframe identifier (ID); used at sites with multiple stations connected to a Cray mainframe or sites with test stations. If this option is omitted, the environment variable $STATID is used. If $STATID is not defined, the default station ID is either ST_DEFIDI or ST_STDEFIDB, both of which are defined in the configuration file **st_hafile**.

-i [*file*]	Initial file the station reads before reading from standard input. If *file* is not specified, but the -i option is specified, the default initial file is identified by the path name set in the environment variable $IASINIT. See subsection 2.2 for more information on environment variables. The first line of the default initial file must be a valid COS ACCOUNT job control statement if accounting is required. The file can also contain commands local to ias. See the ~< interactive station command in the UNIX Station Command Reference Manual for information on redirecting input.
-d *lev*	Debug level; decimal number from 0 to 9. If *lev* is not specified, the debug level defaults to 0. As the debug level increases, more diagnostic messages about the internal state of the interactive station are displayed.
-c *crayid*	Destination Cray mainframe at sites with more than one Cray system. If this option is omitted, the destination defined in the environment variable $CRAY is used. If $CRAY is not defined, the default destination is defined by ST_CRAY in the configuration file st_hafile.

After you enter the ias command line, the system responds with a message and header lines similar to the following:

```
interactive logon -- done
----------------------------------------------------------------------

Use the NEWS. control statement for General CRAY news, Use NEWS,HOURS.
for the CRAY Batch Schedule.  Use NEWS,CLASS. for Job Class information

----------------------------------------------------------------------

mainframe type              mainframe location                    date
COS edition number                                    COS version date

!
```

The header lines of the station interface display may contain a news bulletin followed by the mainframe type, mainframe location, date, COS edition number, COS version level, and assembly date.

If the station or the Cray system is not operating, the system responds with the following message:

```
interactive logon -- Station not responding.
```

If ias attempts to connect to a batch station, the system responds with the following message:

```
Station ID XX is batch.
```

If the interactive logon is successful, the ! prompt appears. You are now logged on to COS as an interactive job and can enter the COS ACCOUNT job control statement if accounting is required. If you enter the COS ACCOUNT job control statement correctly, the ! prompt reappears on the next line. For more information on the ACCOUNT job control statement, see appendix A, Introduction to COS JCL, or the COS Reference Manual.

7.3 USING COS JCL AND INTERACTIVE STATION COMMANDS

After entering the COS ACCOUNT job control statement successfully (if accounting is required), you can enter interactive station commands and COS job control statements. In the following example, the ~s interactive station command requests a job status message from COS.

```
! ~s
COS JOB ABC      IS SUSPENDED                  CP USED=      0 SEC
AB000 - BASE    01301000 LIMIT  01453000 CPU NUMBER   00
```

For more information on the messages returned by COS, see the COS Message Manual.

In the following example, the AUDIT job control statement generates a listing that is placed in the dataset AFILE. Dispose the dataset AFILE to the front-end system by entering the ~d interactive station command.

```
! audit,id=abc,l=afile.
! ~d afile afile.ls
dispose,dn=afile,wait,nrls,mf=UX,text='/usr/username/afile.ls'.
```

The interactive session inserted a batch station identifier (mf=UX) on the DISPOSE statement. Statements not preceded by a tilde (~) are sent directly to COS.

7.4 ENDING A COS INTERACTIVE SESSION

To disconnect your terminal from the COS job, either temporarily or permanently, enter the ~x or ~q station commands, respectively. In the following example, the interactive session is disconnected, but not terminated:

```
    ! ~x
    interactive logoff -- done
    BYE
    $
```

You can log on to the same interactive session again by ending the session with the **~x** command and then entering the ias command. The **~q** interactive station command terminates the job and releases all resources held.

7.5 GETTING HELP WITH THE INTERACTIVE STATION

To request help with any of the interactive station commands, enter **~h**. The following display is returned:

```
    COS interactive station version 3.02 commands:
    ~!               - escape to shell [local]
    ~! cmd           - execute cmd [local]
    ~*               - comment [local]
    ~<[:] file       - redirect input from file to Cray
    ~>[:] file       - redirect Cray output to file
    ~>               - end output redirect
    ~?               - help [local]
    ~a               - attention
    ~b               - abort
    ~c dir           - change directory [local]
    ~d from [to]     - dispose file
    ~e               - send eof
    ~f from [to]     - fetch file
    ~h               - help [local]
    ~n c             - new command character is c [local]
    ~q               - quit (logoff/quit)
    ~s               - status
    ~x               - exit (logoff/hold)
    ~D n             - set debug level n [debug]

    interactive station signals (control characters):
    SIGINT  (CONTROL-C) - interrupt input redirect
    SIGQUIT (CONTROL-\) - terminate (last resort)

    lines that do not begin with the command character are
    blocked and sent to the Cray.
    End help
```

The interactive prompt (!) does not appear after the -h command is issued; however, the station is ready to accept a command.

7.6 SAMPLE INTERACTIVE TERMINAL SESSION

This sample terminal session provides experience in staging files, using the interactive text editor (TEDI), redirecting interactive input and output, and compiling COS interactive jobs.

Before beginning this sample terminal session, you need to prepare a Fortran source file for interactive use on the front-end computer system. After you enter interactive mode, you can do the following:

1. Use the ACCOUNT job control statement to establish your account if accounting is required
2. Stage the Fortran source file to COS and make it local to your job with the -f interactive station command
3. Make the Fortran source file a permanent dataset on Cray mass storage with the SAVE job control statement
4. Use the AUDIT job control statement to display a report on the status of permanent datasets
5. View the source dataset with TEDI
6. Redirect COS output to the screen and to a file on the front-end computer system
7. Compile and load the source dataset by using the CFT77 and SEGLDR job control statements
8. Dispose the source dataset back to the UNIX front end by using the -d interactive station command

To create your source program on the front-end computer system, you must first create a UNIX file named **source** containing the following Fortran statements:

```
         J=50
         DO 10 I=1,10
         J=J+1
         PRINT 50,J
     10  CONTINUE
         STOP
     50  FORMAT(' ',I3)
         END
```

Fortran statements begin in column 7. In the preceding example, Fortran line numbers begin in column 3.

Now that you have created a source file on the front-end computer system, you can begin your interactive session. Enter the ias command after the shell prompt ($).

The system displays the interactive logon message and the site header, followed by the COS interactive prompt (!). You are now logged on to COS as an interactive job and can enter the COS ACCOUNT job control statement (if required) as follows:

```
! account,ac=account,us=userid,upw=userpassword.
!
```

If you enter the ACCOUNT job control statement correctly, the interactive prompt reappears on the next line.

You can now enter the -f station command to stage the source file from the front-end computer system to COS:

```
! -f source
fetch,dn=source,mf=UX,text='/usr/abc/source'.
SS004 - DATASET RECEIVED FROM FRONT END
!
```

If the source file resides in the directory from which the **ias** command was entered, you need only include the simple file name in the TEXT parameter. The -f command concatenates the current directory path name to *fromfile;* however, if the command is typed in long form (**fetch** instead of -f), the default output directory is concatenated to *fromfile*.

The dataset named SOURCE is now local to your COS job. However, this dataset is not a permanent dataset and will not remain on Cray mass storage after your job ends. To make the dataset source a permanent dataset, use the SAVE job control statement:

```
! save,dn=source.
PD000 - PDN = SOURCE       ID =      ED =   1  OWN = U9999
PD000 - SAVE    COMPLETE
!
```

Use the AUDIT job control statement to display the COS permanent dataset status:

```
! audit.
```

The system responds by displaying a report of your permanent datasets known to the system:

```
    ---------------------------   OWN = U9999   --------------------------


    PDN                ID           ED

    SOURCE                          1


          1 DATASET,           2 BLOCKS,           1785 WORDS

    !
```

Because the dataset named SOURCE is now permanent on Cray mass storage, it should be included in your AUDIT report.

You can use TEDI to inspect your source file. Enter the TEDI control statement as follows:

```
    ! tedi,dn=source.
    TE000 - TEDI VERSION    X.15  06/07/85
    TE026 -       1 NULL INPUT RECORDS SKIPPED
     SOURCE    9 LINES
    *
```

In response, TEDI displays messages stating that it has accessed the dataset. To inspect the file, enter T* after the TEDI prompt (*):

```
    *T*
       1      J=50
       2      DO 10 I=1,10
       3      J=J+1
       4      PRINT 50,J
       5  10  CONTINUE
       6      STOP
       7  50  FORMAT(' ',I3)
       8      END
```

TEDI assigns a number to each line of the file. When you are satisfied that the correct file has been staged to COS, exit TEDI mode and return to interactive mode by entering the TEDI END command:

```
    *end
     SOURCE   8 LINES
    !
```

For more information about TEDI, see the Text Editor (TEDI) User's Guide.

You can now compile and load your program interactively. To redirect the output to a file on the front-end computer system, however, you must first enter the following interactive station command:

```
    ! ->cftout
```

This station command redirects the program output to the file **cftout**. The output is also displayed on your screen, unless you specify the colon (:). No interactive prompt is returned after you enter this command. Figure 7-1 shows an example of the output displayed on your screen. To end output redirection, enter the -> interactive station command.

To compile the source dataset, call the Cray Fortran compiler CFT77 with the following command:

```
    ! cft77,i=source.
```

To run a successfully compiled dataset, enter the SEGLDR job control statement:

```
    ! segldr,go.
```

The following output is then displayed on your screen:

```
    LD000 - BEGIN EXECUTION
      51
      52
      53
      54
      55
      56
      57
      58
      59
      60
    UT010 -     STOP            IN $MAIN
```

The Cray Fortran compiler and Cray Assembly Language (CAL) assembler do not rewind the source dataset after compiling or assembling. Therefore, you must rewind the source dataset to recompile or rerun your program interactively. The REWIND job control statement resets the dataset pointer to the first statement in the source dataset. Enter the REWIND job control statement as follows:

```
    ! rewind,dn=source.
```

```
$IN=  0, VM=  0, UM= 96 FIN=  0
CF000 - CFT VERSION -   05/07/85 X.15
1          PAGE 1                ON=CELPQRSTUV              07/31/85-15:32:4
0      CFT X.15(05/07/85)  PAGE 1

      1      1.        J=50
      2      2.        DO 10 I=1,10
      3      3.        J=J+1
      4      4.        PRINT 50,J
      5      5.    10  CONTINUE
      6      6.    50  FORMAT(' ',I3)
      7      7.        STOP
      8      8.        END
1$MAIN     PAGE 2                ON=CELPQRSTUV              07/31/85-15:32
0      CFT X.15(05/07/85)  PAGE 2       $MAIN

OTABLE OF STATEMENT NUMBERS (ALL ADDRESSES IN TABLES ARE OCTAL)
 --------------------------
0  NUMBER USE

     10 SN            50 FN
0  (SN=STATEMENT NUMBER, GSN=GENERATED STATEMENT NUMBER)
   (FN=FORMAT NUMBER, UNDEF*=UNDEFINED STATEMENT NUMBER)
OTABLE OF NAMES ENCOUNTERED (ADDRESS FOR DUMMY ARGUMENT IS THE ARGUMENT NUMBER)
 --------------------------

  ADDRESS NAME     TYPE MAIN USAGE  BLOCK      ADDRESS NAME    TYPE MAIN USAGE  B
LOCK       ADDRESS NAME    TYPE MAIN USAGE  BLOCK

       13 $MAIN     R  ENTRY                           $WFI        EXTERNAL
               11 I         I  VARIABLE
          $STOP        EXTERNAL                        $WFV        EXTERNAL
               10 J         I  VARIABLE
          $WFF         EXTERNAL
0
0 BLOCK NAMES AND LENGTHS IN OCTAL

       37-$MAIN          11-#TB              13-#CL
OSTATIC SPACE (IN OCTAL)
 -----------------------
 B SAVE:               6
 T SAVE:               2
 CONSTANTS:            6
 VARIABLES:            4
 TEMPORARIES:         14
 CODE:                25
               --------
 TOTAL:               63
1          PAGE 1                ON=CELPQRSTUV              07/31/85-15:32:4
0      CFT X.15(05/07/85)  PAGE 3

      9
CF001 - COMPILE TIME =    0.0082 SECONDS
CF002 -        9 LINES,       8 STATEMENTS
CF003 -    52836 WORDS,    18638 I/O BUFFERS USED

!
                                                                      1539
```

Figure 7-1. COS Interactive Job Output

To send the source dataset to the COS output staging queue, use the **~d** interactive station command as follows:

```
! ~d source /usr/abc/tmp/oldsource
dispose,dn=source,wait,nrls,mf=UX,text='/usr/abc/tmp/oldsource'.
!
```

This interactive station command disposes the dataset named SOURCE to the UNIX file named **oldsource** on the front-end computer system.

To end your interactive session, enter the **~x** or **~q** station command.

8. USING UNICOS INTERACTIVELY

This section provides the basic information you need to use the UNICOS interactive facility. This section also includes a sample terminal session to provide you with on-line experience in using UNICOS interactively.

This section discusses the following topics:

- The capabilities of the interactive facility
- Starting a UNICOS interactive session
- Using UNICOS shell commands and interactive station commands
- Ending a UNICOS interactive session
- Getting help with the interactive station

See section 2, Getting Started, for information on setting up your terminal environment. Section 4, Submitting Jobs to UNICOS, describes the UNICOS batch processing facility.

8.1 INTERACTIVE FACILITY OVERVIEW

The interactive facility of the UNIX station allows you to log on interactively to the Cray system to start a UNICOS job. In the interactive environment, you can enter interactive station commands and UNICOS commands one at a time after the default interactive prompt ($ or %; $ is used in this manual) from UNICOS. You can observe the results of each job control statement before you proceed to the next, or redirect output to a specific file on the front-end computer system. For more information on UNICOS commands, see the UNICOS User Commands Reference Manual.

Interactive station commands allow you to perform functions such as transferring files and redirecting interactive input and output. See the UNIX Station Command Reference Manual for a description of the interactive station commands.

8.2 STARTING A UNICOS INTERACTIVE SESSION

You can start an interactive session by entering the **ias** command. When you enter this command, the UNICOS job name defaults to your front-end login name.

```
ias [-l] [-y] [-f char] [-n name] [-b id] [-s statid] [-i [file]]

  [-d lev] [-c crayid]
```

-l	Literal mode; null lines are sent to UNICOS, and trailing blanks are saved on output from UNICOS. By default, each null line is expanded to a single blank on input, and trailing blanks are stripped from output lines.
-y	Synchronous mode; the interactive station waits for a prompt from UNICOS before sending a line of text. By default, input text is sent immediately and queued to UNICOS. Synchronous mode is entered automatically when input is redirected to avoid overrunning UNICOS buffer space.
-f *char*	Character to use for specifying commands local to **ias**. The default is a tilde (~).
-n *name*	UNICOS job name; 1 to 7 characters in length. If *name* is not specified, the default is your front-end login name.
-b *id*	Batch station ID; a 2-character front-end ID of a batch station that will process **acquire**(1), **fetch**(1), and **dispose**(1) requests. This option is applicable only if the batch station is different from the station being used.
-s *statid*	A 2-character station mainframe identifier; used at sites with multiple stations connected to a Cray mainframe or sites with test stations. If this option is omitted, the identifier defined by the environment variable $STATID is used. If $STATID is not defined, the default station ID is either ST_DEFIDI or ST_DEFIDB, both of which are defined in the configuration file **st_hafile**.

-i [*file*] Initial file the station reads before reading from standard input. If *file* is not specified, but the -i option is specified, the default initial file is identified by the path name set in the environment variable $IASINIT. See subsection 2.2 for more information on environment variables. The first line of the default initial file must contain a valid login name and password. The file can also contain commands local to ias. See the ~< interactive station command in the UNIX Station Command Reference Manual for information on redirecting input.

-d *lev* Debug level; decimal number from 0 to 9. If *lev* is not specified, the debug level defaults to 0. As the debug level increases, more diagnostic messages about the internal state of the interactive station are displayed.

-c *crayid* Destination Cray mainframe at sites with more than one Cray system. If this option is omitted, the destination defined in the environment variable $CRAY is used. If $CRAY is not defined, the default destination is defined by ST_CRAY in the configuration file **st_hafile**.

After you enter the ias command line, the system responds with the following:

```
    interactive logon -- done

    login:
    password:
```

You must now enter your UNICOS user ID and password (unless you specify the -i option). The UNICOS prompt appears.

If the station is not operating, the system responds with the following message:

```
    interactive logon -- Station not responding.
```

See subsection 2.2 for a description of the environment variables (such as $IASINIT) required for an interactive session.

8.3 USING UNICOS SHELL COMMANDS AND INTERACTIVE STATION COMMANDS

After successfully entering your UNICOS user ID and password, you can enter interactive station commands and UNICOS shell commands. In the following example, the ~s interactive station command requests a job status message from UNICOS.

```
$ ~s
   F     S    UID     PID    PPID P  PRI    NI   ADDR     SZ     WCHAN TTY   TIME COMMAND
000001 S    342     472     141 0   30    20   4455    141 00052040 p1    0:00 sh
000101 R    342     962     472 0   55    20   4111     72 00000000 p1    0:00 ps
$
```

In the following example, the UNICOS command ls(1) generates a listing that is placed in the file **myfile**. Dispose **myfile** to the front-end system by entering the ~d interactive station command:

```
$ ls > myfile
$ ~d myfile
dispose myfile -mUX -dUD -t'/usr/username/afile.ls'
```

Statements not preceded by a tilde (~) are sent directly to UNICOS.

8.4 ENDING A UNICOS INTERACTIVE SESSION

To disconnect your terminal from the UNICOS job, either temporarily or permanently, enter the **~x** or **~q** station commands, respectively. In the following example, the interactive session is disconnected, but not terminated:

```
! ~x
interactive logoff -- done
bye
$
```

You can log on to the same interactive session again by ending the session with the **~x** command and then entering the **ias** command and using the same user ID. The **~q** interactive station command terminates the job and releases all resources held.

8.5 GETTING HELP WITH THE INTERACTIVE STATION

To request help with the interactive station commands, enter **~h**. The station responds with the display shown in figure 8-1. The interactive prompt (!) does not appear after the ~h command is issued; however, the station is ready to accept a command.

```
interactive station version 3.02 commands:
~!               - escape to shell [local]
~! cmd           - execute cmd [local]
~*               - comment [local]
~<[:] file       - redirect input from file to Cray
~>[:] file       - redirect Cray output to file
~>               - end output redirect
~?               - help [local]
~a               - attention
~b               - abort
~c dir           - change directory [local]
~d from [to]     - dispose file
~e               - send eof
~f from [to]     - fetch file
~h               - help [local]
~n c             - new command character is c [local]
~q               - quit (logoff/quit)
~s               - status
~x               - exit (logoff/hold)
~D n             - set debug level n [debug]

interactive station signals (control characters):
SIGINT  (CONTROL-c) - interrupt input redirect
SIGQUIT (CONTROL-\) - terminate (last resort)

lines that do not begin with the command character are
blocked and sent to the Cray.
End help
```

Figure 8-1. Interactive Help Screen

8.6 SAMPLE INTERACTIVE TERMINAL SESSION

This sample terminal session provides experience in transferring files, using a UNICOS text editor (ed(1)), redirecting interactive input and output, and compiling UNICOS interactive programs.

In preparing for this sample terminal session, you need to have a Fortran source file for interactive use on the UNIX front-end computer system. After you enter interactive mode, you can do the following:

1. Stage the source Fortran file to UNICOS with the ~f interactive station command

2. Use the UNICOS command ls(1) to see if the file has been staged to your directory

3. View the source file with the UNICOS **cat(1)**

4. Redirect UNICOS output to the screen and to a file on the UNIX front-end computer system

5. Compile and load the source file by using the UNICOS commands **cft77(1)** and **segldr(1)**

6. Dispose the source file back to the front end system by using the **~d** interactive station command

To create your source program on the front-end computer system, you must first create a UNIX file named **source** containing the following Fortran statements:

```
      J=50
      DO 10 I=1,10
      J=J+1
      PRINT 50,J
  10  CONTINUE
      STOP
  50  FORMAT(' ',I3)
      END
```

Fortran statements begin in column 7. In the preceding example, Fortran line numbers begin in column 3.

Now that you have created a source file on the UNIX front-end computer system, you can begin your interactive session. Enter the ias command after the shell prompt ($).

The system displays the interactive logon message and the site header, followed by the UNICOS login prompt ($). You are now logged on to UNICOS as an interactive job.

You can enter the **~f** station command to stage the source file from the front-end computer system to UNICOS:

```
$ ~f source
fetch source -fUD -mUX -t'source'
$
```

If the source file and the **ias** command are entered from the same directory (that is, **/usr/abc**), you need only specify the file name in the text field. The **ias** command creates a text field with source name concatenated to the current directory.

The file named **source** should now be in your local directory on UNICOS. To verify this, use the ls(1) command as follows:

```
$ ls
crayjob    example   nqsjob
plan       sample    source
```

You can now use **cat(1)** to inspect your source file. Enter the **cat** command, as follows:

```
$ cat source
8
?
1, 8n

     J=50
     DO 10 I=1,10
     J=J+1
     PRINT 50,J
10   CONTINUE
     STOP
50   FORMAT(' ',I3)
     END
```

When you are satisfied that the correct file has been staged to UNICOS, exit the file and return to interactive mode by entering the **q** command:

```
q
!
```

You can now compile and load your program interactively. To redirect program output to a UNIX file on the front-end computer system, however, you must first enter the following interactive station command:

```
$ ~>cftout
```

This station command redirects the output to the file **cftout**. The output is also displayed on your screen, unless you specify the colon (:). No interactive prompt is returned after you enter this command. To end redirection of output, enter the ~> interactive station command.

To compile the source file, call the Cray Fortran compiler CFT77 with the following command:

```
$ cft77 source.f
```

To run a successfully compiled file, enter the UNICOS command **segldr(1)**:

```
$ segldr source.o
$ a.out
```

The following output is displayed on your screen:

```
a.out
  51
  52
  53
  54
  55
  56
  57
  58
  59
  60
   STOP          in TEST
$
```

To send the source file to the output staging queue, use the **~d** interactive station command as follows:

```
$ ~d source /usr/abc/tmp/oldsource
dispose source -fUD -mUX -t'/usr/abc/tmp/oldsource'
!
```

This interactive station command disposes the file named **source** to the UNIX file named **oldsource** on the front-end computer system.

To end your interactive session, enter the **~x** or **~q** station command.

APPENDIX SECTION

A. INTRODUCTION TO COS JCL

This appendix provides new users with the basic information about the Cray operating system COS necessary to begin using the UNIX station. The following topics are discussed:

- General characteristics of COS
- Dataset characteristics
- Using COS job control language (JCL)

For more detailed information, see the COS Reference Manual.

A.1 COS OVERVIEW

The Cray operating system COS is a multiprogramming, multiprocessing, and multitasking operating system for all Cray systems except a CRAY-2 mainframe. The operating system provides efficient use of system resources by monitoring and controlling the flow of work presented to the system in the form of jobs. COS optimizes resource usage and resolves conflicts when more than one job needs resources.

Jobs are sent to COS through a front-end computer system. In batch mode, you prepare a job file on the front-end system and submit it to COS using the **submit** station command. In COS interactive mode, you log on interactively to COS and enter COS job control or data statements one at a time after the interactive prompt. See section 3, Submitting Jobs to COS, and section 7, Using the COS Interactive Facility, for more information on COS batch and interactive processing.

A.2 DATASET CHARACTERISTICS

A COS dataset, which is similar to a UNIX file, is a quantity of information maintained by the Cray system. All COS datasets are either temporary or permanent. A *temporary dataset* is available only to the job that created it. Temporary datasets are released at the termination of the COS job, unless they are made permanent by use of the SAVE job control statement.

A *permanent dataset* remains on the Cray system after termination of the COS job. Permanent datasets are available to the system and to other jobs and are maintained across system deadstarts. The AUDIT job control statement lists the permanent datasets that users create on the Cray system.

In order to be used by a job, a dataset (temporary or permanent) must be local to that job. Before a permanent dataset can be used by a job, it must first be made local to that job by use of the ACCESS job control statement.

Dataset staging is the process of transferring quantities of information between COS and the front-end computer system. Section 3, Submitting Jobs to COS, and section 5, Transferring Data Between UNIX and COS, discuss transferring data between COS and the front-end computer system.

A.3 USING COS JCL

COS job control language allows you to do the following:

- Identify a job to the system
- Define operating characteristics for the job
- Manipulate datasets
- Call for the loading and execution of user programs
- Call COS programs that perform utility functions for the user
- Define and manipulate other control statements

COS JCL must precede all batch programs submitted to the Cray system. If you prepare your job in a single UNIX file, JCL appears at the top of the UNIX file, above the program code or data (if code or data is included).

COS JCL consists of individual job control statements containing information necessary for a job step to be performed by COS. Generally, job control statements consist of at least a verb and a terminator. In the following example, CFT77 is the verb, and the period (.) acts as a terminator:

```
    CFT77.
```

This job control statement calls the CFT77 compiler into execution.

Most job control statements, however, require parameters and separators that appear between the verb and terminator. Parameters determine how COS understands the statement. They define variables such as dataset names, user identification numbers, and unique access. Separators appear between each keyword/parameter pair or keyword, as in following example:

```
    DISPOSE,DN=TESTJOB.
```

In this job control statement, DN= is a keyword which specifies that the parameter (the dataset TESTJOB) should be staged to the front-end computer system. The comma (,) is a separator. The verb DISPOSE directs a dataset to the COS output queue for staging to a front-end computer system. On the front-end computer system, this file is placed in your default output directory with a name consisting of the COS job name followed by a period and a unique 6-character extension.

Most job control statements have several parameters, each of which defines how the statement is understood by COS. The previous DISPOSE job control statement, for example, can use the SF=*sf* parameter to specify if the dataset will replace or be appended to an existing file on the front-end computer system; the TEXT=*text* parameter can be used to designate the full UNIX path name of the file to which the dataset will be sent. For example:

```
    DISPOSE,DN=TESTJOB,SF=REPLACE,TEXT='/u/abc/tmp/output'.
```

Many job control statement parameters are optional; if you do not include them in your statement, the system supplies default values.

This appendix describes only a few of the parameters available for most of the COS job control statements; the system defaults are used for most values. For a complete description of available parameters, see the COS Reference Manual.

The following subsections describe the job control statements necessary to submit your program to the Cray system. Several of these job control statements are used in the sample terminal sessions in section 7, Using the COS Interactive Facility, and appendix B, COS Sample Terminal Sessions.

A.3.1 IDENTIFYING YOUR JOB - JOB CONTROL STATEMENT

The JOB control statement identifies your job to COS. It is always the first statement in your JCL file, it must begin in column 1, and it cannot be continued onto subsequent lines. As with many job control statements, JOB has several possible parameters, each of which defines how COS understands the statement. Most of the parameters associated with the JOB control statement, however, are optional; if you do not include them in your statement, the system supplies default values.

By taking advantage of default values, you can use the JOB control statement with the following format.

```
 __________________
|                  |
| JOB,JN=jobname.  |
|__________________|
```

JN=*jobname*	Name of the job and its subsequent output; from 1 to 7 alphanumeric characters in length.

For example, the following job control statement identifies a job named testjob to the system.

```
    JOB,JN=TESTJOB.
```

A more detailed format of the JOB control statement is shown here:

```
    JOB,JN=TESTJOB,T=8,P=7,US=U6778.
```

In this example, the time limit for the job is set at 8 seconds, the priority level at which the job enters the system is 7, and the user number is specified as U6778.

A.3.2 VALIDATING YOUR ACCOUNT - ACCOUNT CONTROL STATEMENT

If accounting is mandatory at your facility, you must have an account on the Cray mainframe, and you must include the ACCOUNT control statement in the JCL section of your job file. The ACCOUNT control statement immediately follows the JOB control statement and allows validation of your user number, user password, and account number.

At most installations you do not have to include all available parameters for the ACCOUNT control statement. The following format specifies the information required at most facilities.

```
 ________________________________
|                                |
| ACCOUNT,AC=ac,US=us,UPW=upw.   |
|________________________________|
```

AC=*ac*	Account number; 1 to 15 alphanumeric characters assigned to the user.
US=*us*	User number; 1 to 15 alphanumeric characters assigned to the user.
UPW=*upw*	User password; 1 to 15 alphanumeric characters assigned to the user.

If you are using COS interactively, the ACCOUNT control statement must be the first statement entered in a session. (See section 7 for a discussion of COS interactive processing.)

A.3.3 MINIMUM COS JCL REQUIRED FOR JOB SUBMISSION

This subsection specifies the minimum COS JCL necessary for submitting a batch job to the Cray system. Although you will need to learn additional JCL statements to fully utilize the capabilities of the Cray system, the JCL used in this section allows you to run the sample terminal session in appendix B.

The minimum COS JCL that you must enter with a CFT77 program is shown below:

```
JOB,JN=jobname.
ACCOUNT,AC=ac,US=us,UPW=upw.
CFT77.
SEGLDR,GO.
~e
       First Fortran statement
       .
       .
       .
       Final Fortran statement
~e
```

The first two lines of this example have already been discussed. The fourth line, the SEGLDR control statement, calls the Relocatable Segment Loader into execution. You must enter the ~e indicators before and after the Fortran code to specify the end of the JCL file and the end of the file, respectively.

The minimum JCL for Pascal programs is identical to this example, except that the PASCAL control statement is substituted for the CFT77 control statement. To submit a file containing a Cray Assembly Language (CAL) program, replace the CFT77 control statement with the CAL control statement.

A.3.4 MANAGING PERMANENT DATASETS

After you use the ACQUIRE job control statement (described in section 5), you will need to use the job control statements that allow you to manage permanent datasets stored on the Cray system. This subsection briefly describes the job control statements that enable you to do the following:

- Make an existing permanent dataset local to your COS job (ACCESS)
- Create a record of your own permanent datasets (AUDIT)
- Delete a permanent dataset (DELETE)
- Make a local dataset permanent on the Cray system (SAVE)

A.3.4.1 Access a permanent dataset - ACCESS job control statement

The ACCESS job control statement makes an existing permanent dataset local to a COS job.

```
ACCESS,DN=dn[,PDN=pdn][,ID=uid][,UQ][,OWN=ov][,ED=ed].
```

DN=*dn*	Local dataset name; the name the job will use to refer to the dataset while it remains local to the job. This is a required parameter.
PDN=*pdn*	Permanent dataset name; required if PDN is not the same as DN (for example, if PDN contains lowercase characters).
ID=*uid*	User identification; 1 to 8 alphanumeric characters.
UQ	Unique access; indicates exclusive access to the dataset. UQ is required if you write to, modify, or delete the dataset once it is made local to your job.
OWN=*ov*	Ownership value of the dataset
ED=*ed*	Edition number of the dataset

A.3.4.2 Audit a permanent dataset - AUDIT job control statement

The AUDIT job control statement reports on each of your permanent datasets that exists on the system. When you use AUDIT, a report on the status of permanent datasets appears on the first page of your job output. In its basic form, you can use AUDIT without parameters.

```
AUDIT.
```

AUDIT has several parameters that specify the characteristics of the datasets to be listed. Section 3 discusses the AUDIT report that appears in your job output file.

A.3.4.3 Delete a permanent dataset - DELETE job control statement

The DELETE job control statement removes the permanent status of a dataset on the Cray system. DELETE, however, does not delete a similarly-named dataset on the front-end system. The dataset remains local to the job after the permanent status is deleted.

```
DELETE,DN=dn.
```

DN=*dn*	Local dataset name of the permanent dataset accessed with maintenance permission and unique access

To delete a dataset, you must first gain unique access by using the ACCESS control statement with the UQ parameter.

A.3.4.4 Save a permanent dataset - SAVE job control statement

The SAVE job control statement makes a local dataset permanent on the Cray system.

```
SAVE,DN=dn.
```

DN=*dn*	Local dataset name; the name the job used to create the dataset. This will also be the name that is recorded in the catalog when this form is used.

B. COS SAMPLE TERMINAL SESSIONS

This appendix provides two sample terminal sessions in which you use the UNIX station to prepare, submit, and monitor batch jobs that are sent to COS for processing. These sample terminal sessions illustrate the concepts discussed in sections 2, 3, and 5 of this manual.

Each sample terminal session is divided into three stages:

1. Preparing the job file or job files on the front-end system

2. Submitting the job to COS using the **submit** station command and monitoring the job using other station commands

3. Managing the returned job output on the front end

These three stages are equivalent to the job preparation, submission, and management steps described in section 3, Submitting Jobs to COS, of this publication.

This appendix assumes that you can log on to the front-end system and use a text editor to prepare a job file; if you cannot, see the list of publications in the preface.

These sample terminal sessions provide you with basic models to follow when you prepare, submit, monitor, and manage COS jobs. After you complete these sample terminal sessions, use the UNIX Station Command Reference Manual and the COS Reference Manual to change the COS jobs to meet your requirements.

B.1 TERMINAL SESSION USING THE MINIMUM JCL

In this sample batch terminal session, you will submit a single job file containing the minimum COS JCL necessary for COS processing.

B.1.1 PREPARING THE JOB FILE

The first step in preparing a job for COS processing is to create a UNIX file on the front-end computer system. Use a UNIX text editor to enter the following job control statements and Fortran code into the file named **samplejob**.

```
JOB,JN=CRAYJOB.
ACCOUNT,AC=account,US=userid,UPW=userpassword.
CFT77.
SEGLDR,GO.
~e
      J=50
      DO 10 I=1,10
      J=J+1
      PRINT 50,J
   10 CONTINUE
      STOP
   50 FORMAT(' ',I3)
      END
~e
```

The first four lines of this file contain the minimum JCL required to submit a job that compiles and executes a Fortran program. A description of each line follows.

JOB,JN=*jobname*.
: JOB control statement; specifies the job name you assigned to this job. *jobname* can be from 1 to 7 alphanumeric characters. The Cray computer system returns the job output to your default output directory, using the COS job name, followed by a period and a unique 6-character extension.

ACCOUNT,AC=*account*,US=*userid*,UPW=*userpassword*.
: ACCOUNT control statement; identifies your account number, user ID, and user password to the system. You obtain these values when you open an account with the Cray computer system.

CFT77.
: CFT77 control statement; calls the CFT77 compiler into execution.

SEGLDR,GO.
: SEGLDR control statement; calls the Relocatable Segment Loader into execution.

The end-of-file marker (~e) in line 5 indicates the end of the JCL file. (At some sites, the end-of-file marker may not be ~e). Fortran statements begin in column 7. In this example, Fortran line numbers begin in column 3. A second end-of-file record (~e) indicates the last statement in the file.

After you prepare your UNIX file on the front-end system, you can submit the job for processing on the Cray computer system and monitor its status.

B.1.2 SUBMITTING AND MONITORING THE JOB

To send your job file from the front-end computer system to the Cray computer system, use the **submit** station command followed by the name of the file you are submitting. In this example, you will submit the file **samplejob**, which you previously prepared.

```
    $ cs submit samplejob
```

The system returns the shell prompt ($) after your job has been queued for submission.

To monitor the processing of your job on the Cray computer system, enter the **status** station command. To display only your own jobs, include the **p** operand on the status command line.

```
    $ cs status p
    JOBNAME JSQ   DC  STATUS    CLASS   PRI  FL      CPU      LIMIT   MF  TID
    ------- ----- --  --------  ------- --- -----  --------  -------- --  ---
    CRAYJOB    17 IN  EXECUTE   SMALL     4   13         29       100 UX  abc
                                  End of data
```

To determine if your job is currently executing, look under the STATUS column of the display. In this example, the submitted dataset CRAYJOB is executing, as indicated by the EXECUTE status in the STATUS column.

To display the status of a specific job and its related tasks, use the jstat station command. In the following example, the status of CRAYJOB is displayed because its job sequence number (17) is specified on the jstat command line; CRAYJOB is beginning to execute.

```
    $ cs jstat 17
         JSQ:   17                Cluster:        8     JXT:  00162075
         Job:  CRAYJOB            Priority:       4     JCB:  00500000
        User:                 Field Length:      13     JTA:  00471000
          ID:  UX                Time Used:      29
         TID:  abc              Time Limit:     100
      Status:  EXECUTE      Tapes Reserved:       0
       Class:  SMALL        Tapes Reserved:       0

      $CS:  ldr.
     $LOG:  LD000 - BEGIN EXECUTION

    Tasks:   1      Task #  Status    CP Time          TXT       TCB
                    ------  --------  --------         --------  --------
                         1  EXECUTE         29         00202700  00474647

                                   End of data
```

B.1.3 MANAGING JOB OUTPUT

After COS processing, the job output file is returned to the front-end computer system as described in section 2, Getting Started.

The name of the returned job output file consists of a 1-7 character job name (specified by the COS JOB control statement), followed by a period and a unique 6-character extension. To determine which job output file has been most recently written to your default output directory, use the UNIX ls command with the -t option, as in the following example:

```
    $ ls -t
    CRAYJOB.92348a
    YOURS.93487a
    YOURS.23437a
    MINE.23948a
    $
```

The file most recently written to the directory is at the top of the listing.

See subsection 3.5 for a description of how to read the COS job output file.

B.2 TERMINAL SESSION USING DATASET STAGING

In the first sample terminal session, you submitted a single job file to the Cray computer system. The file contained the minimum JCL required for processing and did not stage files from the front-end computer system or create permanent datasets on Cray mass storage.

This sample terminal session provides you with a more complex example. The primary differences between this terminal session and the first terminal session are as follows:

- The necessary COS JCL and the Fortran program code are stored in two separate UNIX files instead of only one UNIX file. The JCL file is sent to the Cray computer system using the **submit** station command; the Fortran file is staged to the Cray computer system using the ACQUIRE job control statement in the JCL file you will submit.

- The AUDIT job control statement is used to provide a report on the status of the permanent dataset your program stores on the system.

- After the program is executed, a copy of the permanent dataset created on the Cray mainframe is returned to the front-end system by the DISPOSE control statement in your JCL file.

B.2.1 PREPARING THE JOB FILES

For this session, you will prepare two job files on the front-end system. The first file (named **jcljob**) contains the COS JCL; the second file (named **fortran**) contains the Fortran code.

Use a UNIX text editor to enter the following statements into the file **jcljob**.

```
JOB,JN=JCLJOB.
ACCOUNT,AC=account,US=userid,UPW=userpassword.
ACQUIRE,DN=CFTFILE,TEXT='/usr/abc/fortran'.
AUDIT.
CFT77,I=CFTFILE.
SEGLDR,GO.
DISPOSE,DN=CFTFILE.
~e
```

A description of the each line in the file follows.

JOB,JN=*jobname*.
: JOB control statement; specifies the job name you assigned to this job. *jobname* can be from 1 to 7 alphanumeric characters. The Cray computer system returns the job output to your output directory, using the COS job name, followed by a period and a unique 6-character extension.

ACCOUNT,AC=*account*,US=*userid*,UPW=*userpassword*.
: ACCOUNT control statement; identifies your account number, user ID, and user password to the system. You obtain these values when you open an account with the Cray computer system.

ACQUIRE,DN=CFTFILE,TEXT='/usr/abc/fortran'.
: ACQUIRE control statement; stages the UNIX file named **/usr/abc/fortran** from the front end to the Cray computer system, assigning it the local and permanent dataset name of CFTFILE. Later in this JCL file, CFTFILE is compiled by the CFT77 statement and is staged back to the front-end system with the DISPOSE control statement.

AUDIT.
: AUDIT control statement; reports on the status of each of your permanent datasets known to the system. The report should include the permanent dataset CFTFILE, which was created with the previous ACQUIRE control statement.

CFT77,I=CFTFILE.
: CFT77 control statement; calls the CFT77 compiler into execution. I=CFTFILE designates CFTFILE as the dataset from which the compiler will read the Fortran source code.

SEGLDR,GO.
: SEGLDR control statement; calls the Relocatable Segment Loader into execution.

DISPOSE,DN=CFTFILE.
: DISPOSE control statement; stages the local dataset CFTFILE from the Cray computer system to the front end system. To transfer datasets from COS to the front-end system, you must have a user default output directory with other (public) write access.

An end-of-file marker (~e) on the last line of the file indicates the end of the JCL file.

To create the Fortran source file, enter the following code into a UNIX file named **/usr/abc/fortran**. (You may also use the UNIX command **cp** to copy the file from the previous sample terminal session into **/usr/abc/fortran**.)

```
          J=50
          DO 10 I=1,10
          J=J+1
          PRINT 50,J
      10  CONTINUE
          STOP
      50  FORMAT(' ',I3)
          END
    ~e
```

Fortran statements begin in column 7. In this example, Fortran line numbers begin in column 3. An end-of-file record (~e) follows the program and signals the end of the data.

After you have prepared the files **jcljob** and **fortran** on the UNIX front-end system, you can submit them to the Cray computer system for processing.

B.2.2 SUBMITTING AND MONITORING THE JOB

In this sample terminal session, the file **jcljob** (which contains your program JCL) is sent to the Cray computer system; the file **fortran** (which contains your Fortran code) is staged to the Cray computer system by using the ACQUIRE control statement in the file **jcljob**. To submit file **jcljob**, use the **submit** station command as follows:

```
    $ cs submit jcljob
```

The system returns the shell prompt ($) after your job has been queued for submission.

To monitor the processing of your job on the Cray computer system, enter the **status** station command. To display only your own jobs, use the **p** operand.

```
    $ cs status p
    JOBNAME  JSQ   DC  STATUS    CLASS   PRI  FL      CPU      LIMIT   MF  TID
    ------- -----  --  --------  ------- ---  -----  --------  -------- --  ---
    JCLJOB     19  IN  EXECUTE   SMALL     4    13        29       100  UX  abc
                                  End of data
```

To determine if your job is currently executing, look under the STATUS column of the display. In this example, the submitted dataset JCLJOB is executing, as indicated by the EXECUTE status in the STATUS column.

To display the status of a specific job and its related tasks, use the **jstat** station command. In this case, you can request a status display of JCLJOB by including its job sequence number (19) on the **jstat** command line.

```
$ cs jstat 19
     JSQ:   19                 Cluster:     8     JXT:  00162075
     Job:  JCLJOB             Priority:     4     JCB:  00500000
    User:                 Field Length:    13     JTA:  00471000
      ID:  UX                Time Used:    29
     TID:  abc              Time Limit:   100
  Status:  EXECUTE      Tapes Reserved:     0
   Class:  SMALL        Tapes Reserved:     0

  $CS:  ldr.
 $LOG:- LD000 - BEGIN EXECUTION

Tasks:   1      Task #  Status    CP Time        TXT       TCB
                ------  --------  --------       --------  --------
                     1  EXECUTE         29       00202700  00474647
                            End of data
```

The example display shows that the job is beginning to execute.

B.2.3 MANAGING JOB OUTPUT

After your job has been executed by COS, the output is returned to your front-end computer system, as described in section 2, Getting Started. To display your job output file, use a UNIX editor or the UNIX cat command. For information on reading your output file, see subsection 3.5 or the COS Reference Manual.

C. UNICOS SAMPLE TERMINAL SESSIONS

This appendix provides examples you can modify to run your own batch jobs through the UNIX station. The process of batch job submission is divided into three stages:

1. Preparing an NQS job file or job files on the front end system
2. Submitting the job to UNICOS using the **submit** station command and monitoring the job using other station commands
3. Managing the returned job output on the front-end system

Create a batch file consisting of NQS directives and UNICOS shell commands on your front-end system as described in subsection 4.2, Preparing the UNICOS Job File. For example, the following batch job stages a file to UNICOS from the UNIX front-end system:

```
#USER=XX PW=XXXXX
# QSUB-q batch
# QSUB-lT 500
fetch testlib.c -t'/usr/abc/testlib.c'
```

After creating the batch file (**samplejob**), submit the job to UNICOS with the **submit** station command.

```
$ cs submit samplejob
```

The system returns the shell prompt ($) after your job has been queued for submission.

To monitor the processing of your job on the Cray system, enter the **status** station command. To display only your own jobs, include the **p** operand on the **status** command line.

```
$ cs status p
NAME      JSQ   DC  STATUS   CLASS  PRI  FL      CPU   LIMIT  MF  TID
-------  -----  --  -------  -----  ---  --  -------- ------  --  ---
samplej  38894  IN  RUNNING  little   4  13        29     100  UX  abc
```

To determine if your job is currently executing, look under the STATUS column of the display. In this example, the submitted file **samplejob** is running, as indicated by the RUNNING status in the STATUS column.

After UNICOS processing, the job output file is returned to the front-end computer system as described in section 2, Getting Started.

The following examples are included to help illustrate different batch uses of the UNICOS system. For these examples to work on your own system, you must create the appropriate files and specify the mainframe and terminal identifiers associated with your local system. The UNIX command **set** command with the **-x** option displays each command as it is executed. The UNICOS **echo $?** command returns a 0 if the preceding command executed successfully or a nonzero value if the command did not execute successfully.

Example 1:

```
    # USER=xxx PW=xxx
    # QSUB-q batch
    # QSUB-r JOB5
    set -x
    fetch menu -t'/usr/abc/tcs/jobs/big'
    echo $?
    ls -l menu
    echo $?
    date >result.file
    dispose result.file -nRESFILE
    echo $?
    rm menu
```

Example 2:

```
    # USER=xxx PW=xxx
    # QSUB-q batch
    # QSUB-r test2
    set -x
    qstat -l
    ls -l
    sleep 20
    fetch tbox -t'/usr/tcs/tbox'
    echo $?
    cat tbox
    ls -l
    dispose tbox -nBOX1 -dST -t'/usr/abc/tcs/box'
    echo $?
    rm tbox
```

Example 3:

```
    # USER=xxx PW=xxx
    # QSUB-q batch
    # QSUB-r test.job
    set -x
    qstat -l
    ls -l
    cat << EOF > for.f
          program test
          dimension irec(100)
          icount = 10
          do 10 i=1,100
       10 irec(i) = i
          do 30 j=1,icount
          irec(1) = j
          print 20, irec
       20 format (5i10)
       30 continue
          print 40
       40 format (13htest complete)
          stop
          end
    EOF
    cft77 for.f
    segldr for.o
    a.out
    ls -l
    rm for.f for.o a.out
    ls -l
```

Example 4:

```
    # USER=xxx PW=xxx
    # QSUB-q batch
    # QSUB-r cjob
    set -x
    qstat -l
    cat << EOF > c.c
            main() {
            int j=1;
            for (j=1;j<=50;j++) {
                    printf ("j=%d\n",j);
            }
            }
    EOF
    cc -o test.out c.c
    test.out
```

GLOSSARY

A

Abort - To terminate a program or job when a condition (hardware or software) exists from which the program or computer cannot recover

acquire(1) - A UNICOS command that causes the specified dataset to be transferred from the front-end system to the Cray system and saved on the Cray system. This command checks if the file already exists.

ACQUIRE - A COS job control statement that causes the specified dataset to be transferred from the front-end system to the Cray system and made permanent.

Alphanumeric - A character set containing all alphabetic characters (A through Z, $, %, and @) and the digits 0 through 9

Argument - Words that follow the command name on a command line and provide the information necessary for executing a program. Command arguments are usually file names.

ASCII - American Standard Code for Information Interchange

C

Character-blocked (CB) format - Contains ASCII character data, block control words (BCWs), record control words (RCWs), and optional blank compression characters. An end-of-record RCW is interpreted as a line feed or carriage return.

Command - The first word of a command line. It is the name of an executable file that is a compiled program, shell built-in command, or shell procedure.

Command line - A sequence of nonblank arguments separated by blanks or tabs typed by a user. The first word usually specifies the name of a command.

CONTROL-d - Indicates control key sequences. The user holds down the CONTROL key while pressing the terminal key d. Control key sequences can also be shown as a circumflex (^) and a letter, for example ^d, because that is how the system echoes the user entering CONTROL-d. CONTROL-d causes a system interrupt.

COS - The Cray operating system COS

Cray mainframe - The CRAY Y-MP, CRAY-2, CRAY X-MP, CRAY X-MP EA, or CRAY-1 computer system

Current directory - The current point of reference for accessing data within the file system

D

Dataset - A quantity of information maintained on mass storage by the Cray operating system COS. Each dataset is identified by a symbolic name called a dataset name. Datasets are of two types: temporary and permanent. A temporary dataset is available only to the job that created it. A permanent dataset is available to the system and to other jobs and is maintained across system deadstarts.

Deadstart - The process by which an inactive machine is brought up to an operational condition ready to process jobs

Directory - A type of file used to group and organize files and other directories

dispose(1) - A UNICOS command that causes the specified dataset to be transferred from Cray system mass storage to the front-end system

DISPOSE - A COS job control statement that causes the specified dataset to be transferred from the Cray system mass storage to the front-end system

F

fetch(1) - A UNICOS command that causes the specified dataset to be transferred from the front-end system to the Cray system. The file is saved on the Cray system. This command overwrites an existing file.

FETCH - A COS job control statement that causes the specified dataset to be transferred from the front-end computer system to the Cray system. The dataset remains local to the job requesting the dataset.

File - An organized collection of information containing data, programs, or both, which allows users to store, retrieve, and modify information. A simple file name is a sequence of characters other than a slash (/).

Full path name - The path name of a specific file starting from the root directory

I

Input/Output - (1) Commonly called I/O. To communicate from external equipment to the computer and vice versa. (2) The data involved in such a communication. (3) Equipment used to communicate with a computer. (4) The media carrying the data for input/output.

J

Job control statement - Any of the statements used to direct the operating system in its functioning, as compared to data, programs, or other information needed to process a job but not intended directly for the operating system itself. A control statement can be expressed in card, card image, or user terminal keyboard entry medium.

Job output dataset - Any of a set of datasets recognized by the system by a special dataset name (for example, $OUT, $PLOT, and $PUNCH), which becomes a system permanent dataset at job end and is automatically staged to a front-end computer for processing

L

Logoff - A procedure that disconnects the user from UNICOS or COS

Login - A means by which a user can gain access to UNICOS or COS

N

New-line character - The character that appears on a screen as a combination of a carriage return and line feed. To indicate a new-line character on an ASCII keyboard, press the RETURN key. The character is commonly represented in code as \n.

P

Password - A string of up to 13 characters chosen from a 64-character alphabet (., \, 0 through 9, A through Z, a through z)

Path name - A sequence of directory names separated by the / character and ending with the name of a file. The path name defines the connection path between the directory and a file.

R

Relative path name - The path name between the current directory and a specific file

S

Staging - Process of transferring jobs and data in the form of COS datasets and UNIX files from a front-end computer system to Cray mass storage or of transferring datasets from Cray mass storage to a front-end computer system

Standard error - An open file, normally connected to the screen, which receives error messages produced by most commands

Standard input - An open file, normally connected to the keyboard, to which command input is sent

Standard output - An open file, normally connected to the screen, to which output produced by most commands is sent

T

Transparent (TR) format - An ASCII bit string. Data can be in blocked or unblocked format. The end-of-record record control word is ignored and the user must supply carriage control characters.

U

UNICOS - The Cray operating system derived from the AT&T System V operating system. UNICOS is also based in part on the Fourth Berkeley Software Distribution.

V

Variable - A name representing a string value to which the user may assign string values

W

Working directory - See current directory

INDEX

INDEX

READER'S COMMENT FORM

UNIX Station User's Guide SU-0107 C

Your reactions to this manual will help us provide you with better documentation. Please take a moment to check the spaces below, and use the blank space for additional comments.

1) Your experience with computers: ____ 0-1 year ____1-5 years ____5+ years
2) Your experience with Cray computer systems: ____0-1 year ____ 1-5 years ____5+ years
3) Your occupation: ____ computer programmer ____ non-computer professional
   ____ other (please specify): ______________________________
4) How you used this manual: ____ in a class ____as a tutorial or introduction ____ as a reference guide
   ____ for troubleshooting

Using a scale from 1 (poor) to 10 (excellent), please rate this manual on the following criteria:

5) Accuracy _____	8) Physical qualities (binding, printing) ____
6) Completeness ____	9) Readability ____
7) Organization ____	10) Amount and quality of examples ____

Please use the space below, and an additional sheet if necessary, for your other comments about this manual. If you have discovered any inaccuracies or omissions, please give us the page number on which the problem occurred. We promise a quick reply to your comments and questions.

Name ______________________	Address ______________________
Title ______________________	City ______________________
Company ______________________	State/ Country ______________________
Telephone ______________________	Zip Code ______________________
Today's Date ______________________	

CUT ALONG THIS LINE

FOLD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY CARD
FIRST CLASS PERMIT NO 6184 ST PAUL, MN

POSTAGE WILL BE PAID BY ADDRESSEE

CRAY
RESEARCH, INC.

Attention: PUBLICATIONS
1345 Northland Drive
Mendota Heights, MN 55120

FOLD

STAPLE

READER'S COMMENT FORM

UNIX Station User's Guide SU-0107 C

Your reactions to this manual will help us provide you with better documentation. Please take a moment to check the spaces below, and use the blank space for additional comments.

1) Your experience with computers: ____ 0-1 year ____1-5 years ____5+ years
2) Your experience with Cray computer systems: ____0-1 year ____ 1-5 years ____5+ years
3) Your occupation: ____ computer programmer ____ non-computer professional
____ other (please specify): ______________________________
4) How you used this manual: ____ in a class ____as a tutorial or introduction ____ as a reference guide
____ for troubleshooting

Using a scale from 1 (poor) to 10 (excellent), please rate this manual on the following criteria:

5) Accuracy _____	8) Physical qualities (binding, printing) ____
6) Completeness ____	9) Readability ____
7) Organization ____	10) Amount and quality of examples ____

Please use the space below, and an additional sheet if necessary, for your other comments about this manual. If you have discovered any inaccuracies or omissions, please give us the page number on which the problem occurred. We promise a quick reply to your comments and questions.

Name ______________________	Address ____________________
Title ______________________	City ______________________
Company ____________________	State/ Country ______________
Telephone __________________	Zip Code ___________________
Today's Date ________________	

CUT ALONG THIS LINE

FOLD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY CARD
FIRST CLASS PERMIT NO 6184 ST PAUL, MN

POSTAGE WILL BE PAID BY ADDRESSEE

CRAY
RESEARCH, INC.

Attention: PUBLICATIONS
1345 Northland Drive
Mendota Heights, MN 55120

FOLD

STAPLE