UNICOS/mk™ Troubleshooting Technical Note

004–2631–002

New Features

UNICOS/mk™ Troubleshooting Technical Note 004–2631–002

This manual contains the following new or changed information for the UNICOS/mk 2.0.5 release:

- The capability of viewing a live system image using the information server interface has been removed.
- The following directives were added to chapter 2, "`crashmk`":
  - `mem` displays memory size information from the `KnUmkContext` structure.
  - `rlimits` displays resource limits defaults.
- The following changes were made to directives in chapter 2, "`crashmk`":
  - The `-v` option has been added to the `ctx` directive.
  - Aliases `kstk` and `kvs` have been added to the `kstack directive.`
  - The `-d` option has been added to the `proc` directive.
  - The `-c` and `-m` options have been added to the `pwho` directive, and the `-t` option has been removed.
  - The form of the `stack` directive's argument has been expanded.
  - The `-H`, `-P`, and `-t` options have been added to the `status` directive, and the `-v` option has been removed.
  - The `-h` option has been added to the `tsconv` directive.
  - The form of the `utrace` directive's `-b` option argument has changed, and the `-B` and `-L` options have been added.
  - The `-a` option has been added to the `vmlock` directive.

Record of Revision

Version	*Description*
1.4.2	February 1997 Original printing.
1.5	April 1997 Printed to support the UNICOS/mk 1.5 release.
1.6	July 1997 Printed to support the UNICOS/mk 1.6 release.
2.0	October 1997 Printed to support the UNICOS/mk 2.0 release.
2.0.2	January 1998 Printed to support the UNICOS/mk 2.0.2 release.
2.0.3	May 1998 Printed to support the UNICOS/mk 2.0.3 release.
2.0.4	December 1998 Printed to support the UNICOS/mk 2.0.4 release.
002/2.0.5	October 1999 Updated to support the UNICOS/mk 2.0.5 release.

Contents

Page

Figures

Page

Preface

This technical note provides information for troubleshooting crash and dump operations performed by system administrators who manage the operation of any Cray T3E system running the UNICOS/mk operating system.

Related Publications

The following documents contain additional information that may be helpful:

- *UNICOS/mk General Administration*
- *UNICOS/mk Resource Administration*
- *UNICOS/mk Configuration Reference Manual*
- *UNICOS/mk Networking Facilities Administration*
- *Tape Subsystem Administration*
- *UNICOS/mk Installation Guide for Cray T3E Series Systems*
- *SWS-ION Release Overview*
- *SWS Solaris Operating System and Devices Installation Guide*
- *SWS-ION Administration and Operations Guide*
- *Cray Scalable I/O Functional Overview*
- *Cray Scalable I/O Messages*

Obtaining Publications

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

To order a document, call +1 651 683 5907. SGI employees may send e-mail to `orderdsk@sgi.com`

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

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

Conventions

The following conventions are used throughout this document:

Convention	Meaning
`command`	This fixed-space font denotes literal items (such as commands, files, routines, pathnames, signals, messages, programming language structures, and e-mail addresses) and items that appear on the screen.
`manpage`(*x*)	Man page section identifiers appear in parentheses after man page names. The following list describes the identifiers:

1	User commands
1B	User commands ported from BSD
2	System calls
3	Library routines, macros, and opdefs
4	Devices (special files)
4P	Protocols
5	File formats
7	Miscellaneous topics
7D	DWB-related information
8	Administrator commands

Some internal routines (for example, the `_assign_asgcmd_info`() routine) do not have man pages associated with them.

Convention	Meaning
variable	Italic typeface denotes variable entries and words or concepts being defined.
`user input`	This bold, fixed-space font denotes literal items that the user enters in interactive sessions. Output is shown in nonbold, fixed-space font.

[ ]	Brackets enclose optional portions of a command or directive line.
...	Ellipses indicate that a preceding element can be repeated.

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

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

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

Reader Comments

If you have comments about the technical accuracy, content, or organization of this document, please tell us. Be sure to include the title and part number of the document with your comments.

You can contact us in any of the following ways:

- Send e-mail to the following address:

  `techpubs@sgi.com`

- Send a fax to the attention of “Technical Publications” at: +1 650 932 0801.
- Use the Feedback option on the Technical Publications Library World Wide Web page:

  `http://techpubs.sgi.com`

- Call the Technical Publications Group, through the Technical Assistance Center, using one of the following numbers:

  For SGI IRIX based operating systems: 1 800 800 4SGI

  For UNICOS or UNICOS/mk based operating systems or Cray Origin 2000 systems: 1 800 950 2729 (toll free from the United States and Canada) or +1 651 683 5600

- Send mail to the following address:

Technical Publications
SGI
1600 Amphitheatre Pkwy.
Mountain View, California 94043–1351

We value your comments and will respond to them promptly.

Introduction to UNICOS/mk Troubleshooting [1]

This technical note is for people who manage the operation of Cray Research Cray T3E computer systems running the UNICOS/mk operating system and who have a large amount of experience with the UNICOS/mk operating system.

Information in this technical note is highly informal and changes often.

> **Note:** This document does not answer all questions or solve all problems. If you need assistance in debugging a system crash, or if you have a recurring UNICOS/mk kernel problem, contact your local Cray Research support personnel. When you call, it is important to have available dumps and a copy of the UNICOS/mk operating system that caused the crash.

This technical note provides the following information:

- A description of `crashmk`(8) command conventions.
- Detailed descriptions of the `crashmk`(8) command and its directives.
- Diagrams of data structures used or displayed by selected `crashmk`(8) directives, and notes on the interpretation of selected directives' output.

This technical note does not replace experience nor other documents that more fully describe specific system areas. To effectively manage a Cray T3E system running the UNICOS/mk operating system, both familiarity with the references listed in the preface and the full-time effort of an individual is necessary.

`crashmk` [2]

The `crashmk`(8) utility is used to examine an operating system image. It has facilities for displaying and interpreting control structures, traces, logs, and so forth.

2.1 Conventions

The following conventions are used to identify instances where there are alternatives in portions of the syntax of a command or directive.

- A set of alternatives is bounded by **{** and **}**.
- Within these bounds the alternatives are delimited by **|** or **||**.
  - If the delimiter is **|**, any or all of the alternatives may be selected.
  - If the delimiter is **||**, the alternatives are mutually exclusive and only one may be selected.

2.2 `crashmk` Command Line

The format of the `crashmk` command line follows:

```
crashmk [-a offset_file] [-b pe#] [-C maxopen] [-D] [-DD] [-i dirfile]
   [-I] [-m mode] [-n namelist] [-o outfile] [-p num] [-s] [-U] [-x pe#]
   [system_image [namelist] ]
```

`-a` *offset_file*	Specifies file containing memory address offset specifications. See Section 2.2.3, page 7, for a description of the specifications.
`-b` *pe#*	Specifies the processing element (PE) number of the boot PE. The boot PE becomes the initial target PE if `-x` *pe#* is not specified. See Section 2.2.4, page 8, and Section 2.2.5, page 9.

`-C` *maxopen*	Specifies the maximum number of PE system images that may be open simultaneously. If specified, *maxopen* must be 3 or larger. The default value is 10.
`-D`	Enable debug mode excluding processing the *namelist* file. See `debug` directive, Section 2.3.2.18, page 26.
`-DD`	Enable debug mode including processing the *namelist* file. See `debug` directive, Section 2.3.2.18, page 26. **Note:** A line of debug output is produced for each symbol in the *namelist* file. Redirecting output to a file in silent mode, specified by the `-o` and `-s` options, is recommended.
`-i` *dirfile*	Specifies a file containing directives. Directives are read from *dirfile* until either an end of file condition or a `quit` directive is encountered, after which `crashmk` exits. The same effect can be obtained by using a pipe or by input redirection, as in the following examples. `cat` *dirfile* `|` `crashmk` *options args* — or — `crashmk` *options args* `<` *dirfile*
`-I`	Uses *new input style.* The number base of addresses specified in directives is determined using the *new* input style. See Section 2.3.1.2, page 12.
`-m` *mode*	Specifies default display mode. See Section 2.3.1.3, page 12, for valid *mode* values.
`-n` *namelist*	Specifies the name of the symbol information file that corresponds to the system images being processed. See Section 2.2.2, page 6.

`-o` *outfile*	Redirects output to *outfile.* To terminate redirection within `crashmk`, use `redirect` with no file name.
`-p` *num*	Specifies the number of PEs. See Section 2.3.2.18, page 26.
`-s`	Silent mode — sends output to *outfile* only. Ignored if `-o` option is not specified.
`-U`	Unconditional. Do not abort if: • *namelist* is not specified and no default file is found. • The specified *namelist* file cannot be opened.
`-x` *pe#*	A hexadecimal number that specifies the initial target PE. **Note:** This option is effective only when *system_image* specifies a directory or a live system.
system_image	Specifies where the system image can be found. This argument can have one of the following values: • `/dev/pe`, a UNICOS/mk character special device, which can be used to view the system image of a live system. • Name of a file that contains a system image. See Section 2.2.1, page 6. • Name of a directory containing one or more system image files. If `-x` *pe#* is specified and the file `dump.`*pe#* exists, that file is the initial target PE. Otherwise, the system image file of the lowest numbered PE becomes the initial target PE. See Section 2.2.1, page 6. • Socket ID, of the form *host:port#.* This form is used to view the system image of a live system

via the `skdb` server running on *host*, and listening on port *port#* (*host* is generally not the host on which the system image is running).

- Dash character (`---`) which serves as a place holder when *namelist* is specified as a positional parameter, but no *system_image* specification is wanted.

  Note: The preferred method of specifying *namelist* is with the `-n` option.

- If omitted and `crashmk` is running under the UNICOS/mk operating system, the default is `/dev/pe`.

  Note: There is no default when `crashmk` is not running under the UNICOS/mk operating system.

namelist

Specifies the name of the symbol information file that corresponds to the system image(s) being processed. If *namelist* is specified by both the `-n` option and by this positional parameter, the `-n` option specification takes precedence.

See Section 2.2.2, page 6.

2.2.1 System Image File Names

System image file names are expected to be of the form `dump.`*xxx*, where *xxx* is the 3-digit hexadecimal number that corresponds to the PE number of the dumped PE.

2.2.2 The *namelist* File

The default *namelist* values, in order of search, are: `unicosmk.cray-t3e~`, `unicosmk.cray-t3e`, `unicosmk~`, `unicosmk`, `unicosmk.ar~`, and `unicosmk.ar`.

If the *namelist* argument is not specified, the following locations are searched for a default *namelist* file:

1. Directory that contains *system_image*, if specified

2. Current directory

If `crashmk` is executed on a Cray T3E system, the following locations are searched in addition to those above:

1. `/dumps/current` directory
2. The root directory (`/`)

If no *namelist* file can be opened, or no *namelist* file containing symbol information is found, the actions taken are:

1. If the `-U` option was specified on the command line, continue.
2. If input is from a tty device, ask the user whether to continue or exit, and act accordingly.
3. Exit.

2.2.3 Offsets File

The offsets file is a text file that contains memory offset adjustment specifications. This file is used in cases where the memory image of a PE has been shifted and is read by `crashmk` during initialization.

The format of the specification line is:

```
PE-#  memory_address  offset_adjustment
```

The fields are:

PE-#	PE number, assumed to be hexadecimal.
memory_address	Memory base address, in bytes. The adjustment will be applied to read addresses equal to or greater than the base address, but less than the address in the preceding specification. The address is interpreted in the same manner as a directive address specification. See Section 2.3.1.1, page 10, for details on address specification.
offset_adjustment	Offset value, in bytes. When memory is read the seek address is computed by subtracting the offset

from the read address. A negative adjustment is specified by a minus sign (-) preceding the value.

- Specification lines must be in descending *memory_address* order within PE numbers.
- The fields are separated by at least one white space character (tab or space).
- Leading and trailing white space is ignored.
- Lines in which the first nonwhite space character is a pound sign (#) are assumed to be comments and are ignored.

For example, if memory beginning at address `0x1bffc0` on PE `0x00f` had been shifted downward by `0xbb0` bytes, the specification line would be:

```
0x00f 0x1bffc0 0xbb0
```

Note: The preceding information is available from within `crashmk`(8) via the `help offsets` directive.

2.2.4 Boot PE and Number of PEs

The boot PE is the first PE that is started during a system boot. It controls the loading, initialization, and configuration of the remaining PEs. The configuration server (`config`) is always located on the boot PE. The boot PE number is set in the configuration file on the SWS prior to booting the system.

During the boot process the boot PE number and the number of PEs are written by the SWS into a fixed memory location (the boot information block, `typedef boot_info_t`) in the boot PE. During initialization a copy of the boot information block is then written to each of the other PEs by the boot PE.

It is important that `crashmk` has access to the boot PE, as a number of its directives use information from the configuration server's tables.

The number of PEs is needed as a limit by directives that scan PEs. Generally, `crashmk` is able to get these values from the boot information block of the initial target PE. However, if the boot information block has been corrupted, it may be necessary to specify the boot PE and/or the number of PEs manually. This is best done by exiting `crashmk`, then re-executing it, specifying the boot PE and/or the number of PEs using the `-b` and `-p` options, respectively.

2.2.5 Target PE

During the course of processing a directive, `crashmk` may switch between PEs. A PE that is being switched to is called the *target PE*.

The *initial target PE* is the PE that is open when `crashmk` prompts for the first directive. It can be explicitly specified by the `-x` option, or implied by the `-b` option. If neither the `-b` nor `-x` options are specified, the default is the boot PE.

> **Note:** When the default is chosen, `crashmk` selects a PE to open, usually PE-0, and gets the boot PE number from the boot information block. Then it switches to the boot PE, if necessary.

2.2.6 Environment Variables

The following environment variables can be used to set default values:

`CRASH_DEFMODE` *mode*

Specifies default display mode.

See Section 2.3.1.3, page 12, for valid *mode* values.

If default display mode is specified by both `CRASH_DEFMODE` and the `-m` option, the `-m` specification takes precedence.

`CRASH_NEWINPUTSTYLE [TRUE]`

If set to `TRUE`, the number base of addresses specified in directives is determined using the *new* input style.

See Section 2.3.1.2, page 12.

2.2.7 Exit Values

When `crashmk` exits, one of the following values is returned:

0	No error.
1	An error occurred during the most recent directive's processing.
2	A failure occurred while reading or processing the symbol information file, *namelist*.
3	A failure occurred while reading or processing the system image.

4	An error occurred other than one of the previous types.

2.3 `crashmk` Directives

2.3.1 Format

Directives are generally of the form:

```
directive [options] [arguments] [{ >file || >> file || | command }]
```

directive	Identifies the operation to be done.
options	Modifies the operation and/or the output. The `-D` option enables *directive-level* debug mode, and can be specified with any directive. The effect is as if the directive had been proceeded by the `debug on` directive, and followed by the `debug off` directive. See the `debug` directive, Section 2.3.2.18, page 26. In cases where an option is repeated, the effects are cumulative. For example, a specification of `-ll` means that the effects described for the `-l` option occur along with the effects described for the `-ll` option.
arguments	Addresses, counts, slot numbers, and so on. See Section 2.3.1.1, page 10, for details on address specification.
> *file*	Writes the directive's output to *file.*
>> *file*	Appends the directive's output to *file.*
| *command*	Pipes the directive's output to *command.*

2.3.1.1 Address Specification

Addresses can be specified as either a numeric value, a symbol, or a table or pool entry. The forms are:

```
{ [*]numeric_value || [*][actor.]symbol || name[[slot#]] }
```

The form is determined as follows:

1. If the character `[` is present, the address is presumed to be that of a table or pool.

   Note: The characters `[` and `]`are part of the form and must be specified.

   - If *slot#* is present and is numeric, the address is that of the table or pool entry specified by *slot#*.
   - If *slot#* is omitted and *name* is that of a table, the address is that of the beginning of the table array - equivalent to *table* `[0]`.
   - If *slot#* is omitted and *name* is that of a pool, the address is that of the pool's index array.

     In this case, the following steps are omitted.

2. If the first character is within the range `0` through `9`, or if the prefix `0x` is present, the address is presumed to be numeric.

   Note: If the first character is one of the hexadecimal digits `a` through `f`, the prefix `0x` must be specified for it to be interpreted as a numeric value.

3. Otherwise, the address is presumed to be a symbol.

4. If the first character is the indirect addressing indicator, `*`, the tests in steps 2 and 3 are applied to the second character. When indirect addressing is specified, the **content** of the memory location at the specified address is used as the address.

The rules used to determine the number base of numeric value addresses depend on the input style setting. See Section 2.3.1.2, page 12.

Symbols are interpreted as follows:

actor	Specifies the actor (server) within which *symbol* is defined. If omitted, the first occurrence of *symbol* is used.
symbol	A symbol, the value of which is used as the address.

Symbols of the form *actor.symbol* are qualified symbols, where *actor* is the qualifier. Symbols of the form *symbol* are unqualified symbols.

Note: The preceding information is available from within `crashmk` via the `help addrform` directive.

2.3.1.2 Input Style for Addresses

Traditional and new input styles for address specification are available in `crashmk`. By default, `crashmk` uses the traditional style.

In the traditional input style, the number base (octal, decimal or hexadecimal) of addresses that are arguments of `crashmk` directives is defined by the `default mode` directive — initially hexadecimal. The exception being, addresses with the prefix `0x` are always interpreted as hexadecimal.

In the new input style, the number base of addresses that are arguments of `crashmk` directives is defined explicitly by their prefix: `0` specifies octal, `0x` specifies hexadecimal, and decimal otherwise.

The new input style is enabled by the `-I` option on the `crashmk` command line, or by setting the environment variable `CRASH_NEWINPUTSTYLE` to `TRUE`.

For example, in the Borne or Korn shells:

```
CRASH_NEWINPUTSTYLE=TRUE
export CRASH_NEWINPUTSTYLE
```

Or in the C shell:

```
setenv CRASH_NEWINPUTSTYLE TRUE
```

Note that the number base of numeric arguments that are **not** addresses (for example, entry numbers, slot numbers, counts, and so on) is not affected by the input style for addresses. The number base of these types of arguments is always determined by their prefix - `0` for octal, `0x` for hexadecimal, otherwise decimal.

Note: The preceding information is available from within `crashmk` via the `help style` directive.

2.3.1.3 Mode Settings

The *mode* setting defines the number base in which non-decimal numeric output is displayed. When traditional input style is used, it defines the number base of numeric addresses that are arguments of directives. It also defines the default output format of the `od` directive.

The following table shows the mode value, number base, and description of output when specified as *format* on the `od` directive (section Section 2.3.2.47, page 46). When *last setting* appears in the Number Base or `od` Output columns, it means that the Mode Value does not change the number base, and the last setting, octal or hexadecimal, is remains in force.

Table 1. Mode Settings

Mode Value	Number Base	`od` Output
`octal`	octal	octal
`oct`	octal	octal
`o`	octal	octal
`hexadecimal`	hexadecimal	hexadecimal (lowercase `a` through `f`)
`HEXADECIMAL`	hexadecimal	hexadecimal (uppercase `A` through `F`)
`hexadec`	hexadecimal	hexadecimal (lowercase `a` through `f`)
`HEXADEC`	hexadecimal	hexadecimal (uppercase `A` through `F`)
`hex`	hexadecimal	hexadecimal (lowercase `a` through `f`)
`HEX`	hexadecimal	hexadecimal (uppercase `A` through `F`)
`h`	hexadecimal	hexadecimal (lowercase `a` through `f`)
`H`	hexadecimal	hexadecimal (uppercase `A` through `F`)
`x`	hexadecimal	hexadecimal (lowercase `a` through `f`)
`X`	hexadecimal	hexadecimal (uppercase `A` through `F`)
`och`	octal	octal with character interpretation
`hch`	hexadecimal	hexadecimal (lowercase `a` through `f`) with character interpretation
`HCH`	hexadecimal	hexadecimal (uppercase `A` through `F`) with character interpretation
`character`	*last setting*	character
`char`	*last setting*	character
`c`	*last setting*	character
`byte`	*last setting*	byte, *last setting*
`b`	*last setting*	byte, *last setting*
`decimal`	decimal	signed decimal

Mode Value	Number Base	`od` Output
`dec`	decimal	signed decimal
`d`	decimal	signed decimal
`udec`	decimal	unsigned decimal
`ud`	`udec`	unsigned decimal
`D`	`udec`	unsigned decimal
`iw`	*last setting*	*last setting*, reversed half-word DEC Alpha instruction layout

Note: The preceding information is available from within `crashmk` via the `help mode` directive.

2.3.1.4 Name Mangling

The C++ language enforces strong type checking of function linkages. This is accomplished by encoding information about the number and types of arguments within the internal representation of the function's name. This practice is referred to as *name mangling*, and an encoded function name is referred to as a *mangled name*. For example, if a function declaration is:

```
Exec(tPmProc*,tPmUThread*,tPmExeceArg*,int)
```

The *mangled name* is:

```
Exec__FP7tPmProcP10tPmUThreadP11tPmExeceArgi
```

Function and variable names are *demangled* by `crashmk` during the processing of symbol entries that occurs at start up. Names displayed by directives such as `ds` or `stack` are *demangled*.

2.3.1.5 Range Format

Some arguments can be expressed as ranges. A *range* comprises a starting number, an ending number, and all numbers in between. The numbers can be decimal integers (typically slot or entry numbers), or hexadecimal PE numbers.

The general form is *start*[-*end*]. If *end* is not specified, the default value is *start*. If *start* is less than *end*, the processing order is ascending. Otherwise, it is descending.

The keyword `bootpe`, an alias for the boot PE number, can be used as a starting or ending value for PE ranges.

PE names which are specified in the configuration file (for example, `pe_name = "ospe_a"`) can be used as starting or ending values of PE ranges.

> **Note:** When PE names are specified, the PE where the `config` server must be present in the system image.

The following keywords, which imply ranges, can be used in place of numeric values. In this case the form is simply *keyword*.

Table 2. Integer Range Keywords

Keyword	Implied range
`all`	all entries (slots)

Table 3. PE Range Keywords

Keyword	Implied range
`all`	all PEs
`apppe`[s]	all application PEs
`cmdpe`[s]	all command PEs
`ospe`[s]	all OS PEs

When keywords are used, the processing order of PE ranges depends on the PE number of the boot PE. If the boot PE is 0, the order is ascending; otherwise, it is descending.

> **Note:** The preceding information is available from within `crashmk` via the `help range` directive.

2.3.2 Directives

2.3.2.1 !

Use `fork(1)` to create a shell and execute the indicated command.

```
! command [; command ...]
```

Aliases: NONE

2.3.2.2 .

Repeats the previous directive.

```
.
```

Aliases: NONE

Fork requests (e.g., `! ls`) are not repeated. If the `.` directive follows a fork request, the last non-fork directive is repeated.

2.3.2.3 actor

Display actor information.

```
actor [-e] [-k] [-l] [-n] [-p] [-q] [-r] [-t] [-v]
   [ [pe_range:][slot_range] ...]
```

Alias:	`a`
`-e`	Display all actors on all sites.
`-k`	Include stack traces for threads of specified actors.
`-l`	Include actor details. If the [`-p`] and/or [`-t`] options are specified, process and/or thread details are included.
`-n`	Include the name of the nearest symbol or pool for stack arguments that appear to be addresses. This option is ignored if the `-k` option is not specified.
`-p`	Include information about the process associated with the specified actor, if any.
`-q`	Include swap information. If specified, the `-l` option is implied.
`-r`	Include the stacks of threads on remote sites that either created, or were created by the actor's thread. This option is ignored if the `-t` option is not specified.

`-t`	Include thread information.
`-v`	Include idle threads. This option is ignored if the `-t` is not specified.
pe_range	A range of PE numbers. Numbers are assumed to be hexadecimal, and the trailing colon (`:`) is required. The default is the current PE.
slot_range	A range of actor entry (slot) numbers.

See Section 2.3.1.5, page 14, for details of the range format.

By default, a summary of all actors on the current PE is displayed.

The `-e` option is ignored if arguments are specified.

If the `-t` option is specified along with the `-l` option, thread details are included.

2.3.2.4 address

Display the address of a table entry.

```
address name slot#
```

Aliases:	`addr`, `addrof`
name	Name of a table or pool (for example, `Actor`, `file`, etc.).
slot#	Table or pool entry (slot) number.

name`[`*slot#*`]` is accepted as an alternate form (for example, `Proc[2]`).

If *name*`[]` is specified and *name* is the name of a table, the address of the beginning of the table array (equivalent to *name*`[0]`), the number of entries, and entry size are displayed. For example:

```
0x217> addr file[]
  file table address: 0x1800040, #-entries: 8192, entry size: 136 (17 wd)
```

If *name*`[]` is specified and *name* is the name of a pool, the address of the pool's index array, the number of entries, and entry size are displayed. For example:

```
0x21c> addr thread[]
  &Thread index array = 0x730040, #-entries = 128, entry size: 712 (89 wd)
```

2.3.2.5 `bits`

Display the ordinal number of the set bits in a value.

```
bits [-z] value ...
```

Aliases:	NONE
`-z`	Display the bits that are not set.
value	Number to be evaluated.

Note: The right-most bit of *value* is bit-0.

2.3.2.6 `buffer`

Display the contents of system buffers.

```
buffer [-m mode] [-w] [pe_range:]slot_range ...
```

Alias:	`b`
`-m` *mode*	Specifies the display mode. The initial default is the current mode as specified by the `default mode` directive. After *mode* is specified, it becomes the default for subsequent `buffer` directives, until overridden. Any valid mode can be specified. See Section 2.3.1.3, page 12, for valid *mode* values.
`-w`	Include a list of PEs where a file server is loaded.
pe_range	A range of PE numbers. Numbers are assumed to be hexadecimal, and the trailing colon (`:`) is required. The default is the current PE if it has a file server. Otherwise, the default is the root file server's PE. Numbers of PEs that do not have file servers are silently ignored.
slot_range	A range of system buffer header entry (slot) numbers.

See Section 2.3.1.5, for details of the range format.

System buffers are found only on PEs with a file server.

If the `-w` option is specified without arguments, only file server information is displayed. Otherwise, if no arguments are specified, the default is to display all in use entries on the default PE.

Arguments are ignored when the `-e` option is specified.

2.3.2.7 `bufhdr`

Display system buffer headers.

```
bufhdr [-a] [-e] [-l] [-w] [ [pe_range:][slot_range] ...]
```

Aliases:	`buf`, `hdr`
`-a`	Include headers having a device type of `NODEV` (not in use). This option is assumed when arguments are specified.
`-e`	Display all in use headers on all PEs where system buffer headers exist.
`-l`	Include links.
`-w`	Include a list of PEs where a file server is loaded.
pe_range	A range of PE numbers. Numbers are assumed to be hexadecimal, and the trailing colon (`:`) is required. The default is the current PE if it has a file server. Otherwise, the default is the root file server's PE. Numbers of PEs that do not have file servers are silently ignored.
slot_range	A range of system buffer header entry (slot) numbers.

See Section 2.3.1.5, page 14, for details of the range format.

System buffer headers are found only on PEs with a file server.

If the `-w` option is specified without arguments, only file server information is displayed. Otherwise, if no arguments are specified, the default is to display all in use entries on the default PE.

Arguments are ignored when the `-e` option is specified.

2.3.2.8 `cdcmp`

Compares a code segment (the suspect code) with the corresponding segment (the target code) in either the *namelist* file or in another PE.

```
cdcmp [-l length] [-n] [-p target_pe] [-s num] [-v] [ [pe#:]code_seg ...]
```

Aliases:	`ccmp`, `code`, `codecmp`
`-l` *length*	The length, in **words**, of the code to be compared. The prefix `b:` or `B:` can be used to indicate a byte value. When *code_seg* is a server name, the default is the size of the server segment. When *code_seg* is an address, the default is 512 words.
`-n`	Include the name of the nearest symbol or pool for locations where a mismatch is found.
`-p` *target_pe*	Specifies the PE number, in hexadecimal, of the target code. The keyword `bootpe` can be used to specify the boot PE. The default is the *namelist* file.
`-s` *num*	Stops displaying mismatches after the first *num* have been displayed. If the keyword `all` is specified, all mismatches are displayed. The default is 30.
`-v`	Display all code. The default is to display mismatches only.
pe#	The PE number. Numbers are assumed to be hexadecimal, and the trailing colon (`:`) is required. The default is the current PE.
code_seg	Specifies the code segment to be compared. Can be either a server name or an address.

- For server name:

  server[`,` *type*]

  Where, *server* is the name of a server, and *type* is the segment type. Valid *type* specifications are `cd` (code), or `da` (data). The default is `cd`. The keyword `all` can be specified for *server*. In this case the *type* segments of all servers on *pe#* are compared.

- For addresses:

  addr The starting address of the code to be compared.

The default is all servers on the current PE.

If the `-s` and `-v` options are specified, the `-v` option takes precedence, and all code is displayed.

2.3.2.9 `chtab`

Display NFS client handle entries.

```
chtab [-a] [-e] [-l] [-w] [ [pe_range:]entry ...]
```

Aliases:	`chtable`, `cht`, `ch`
`-a`	Include unused entries. This option is assumed when arguments are specified.
`-e`	Display all in use entries on all PEs where a client handle table exists.
`-l`	Generates a long form report that includes the `mountinfo` and the `RPC CLIENT` structures.
`-w`	Include a list of PEs where a file server is loaded.
pe_range	A range of PE numbers. Numbers are assumed to be hexadecimal, and the trailing colon (`:`) is required. The default is the current PE if it has a file server. Otherwise, the default is the root file server's PE. Numbers of PEs that do not have file servers are silently ignored.
entry	Client handle table entry. Can be either of the following:

slot_range	A range of client handle entry (slot) numbers.
address	Address of a client handle entry. The address is ignored if it is not within the client handle table. Otherwise, the entry nearest to the address from below is displayed.

See Section 2.3.1.5, page 14, for details of the range format.

Client handle tables are found only on PEs with a file server.

If the `-w` option is specified without arguments, only file server information is displayed. Otherwise, if no arguments are specified, the default is to display all in use entries on the default PE.

Arguments are ignored when the `-e` option is specified.

2.3.2.10 `cku_private`

Display NFS `cku_private` structures.

```
cku_private address ...
```

Aliases:	`cku_priv`, `ckup`
address	Structure address. Data at the specified address will be formatted as if it were a `cku_private` structure.

2.3.2.11 `cm_conn`

Display DFS `cm_conn` structures.

```
cm_conn [-g] [-l] [-s] [-v[v[v[v]]]] [address ...]
```

Alias:	`cmc`
`-g`	Display output in a form suitable for a search using `grep`(1).
`-l`	Include locking and usage information.
`-s`	Include `cm_server` information.
`-v`	Increase level of output detail.
address	Address of a `cm_conn` structure. By default, all `cm_conn` structures are displayed.

`cm_conn` structures are found in the root file server only when DFS is configured.

2.3.2.12 `cm_serv`

Display DFS `cm_server` structures.

```
cm_serv [-c] [-g] [-l] [-v[v[v[v]]]] [address ...]
```

Alias:	`cms`
`-c`	Include `cm_conn` information.
`-g`	Display output in a form suitable for a search using `grep`(1).
`-l`	Include locking and usage information.
`-v`	Increase level of output detail.

address	Address of a `cm_server` structure. By default, all `cm_server` structures are displayed.

`cm_server` structures are found in the root file server only when DFS is configured.

2.3.2.13 cmon

Display C-option performance monitor array information.

```
cmon [-T]
```

Aliases:	NONE
`-T`	Display the time stamps in raw form.

The amount of information is substantial. Redirecting the output to a file is recommended.

2.3.2.14 cnode

Display GigaRing `cnode` structures.

```
cnode [-e] [-w] [pe_range ...]
```

Alias:	`cn`
`-e`	Display `cnode` structures on all PEs where a packet server is loaded.
`-w`	Include a list of PEs where a packet server is loaded.
pe_range	A range of PE numbers. Numbers are assumed to be hexadecimal. The default is the current PE if a packet server is loaded.

`Cnode` structures are found only on PEs with a packet server.

If the `-w` option is specified with neither the `-e` option nor arguments, only packet server information is displayed. Otherwise if no arguments are specified, the default is to display `cnode` structures on the current PE.

Arguments are ignored when the `-e` option is specified.

2.3.2.15 `config`

Display configuration server information.

```
config -p [-a] [-A] [-C] [-l] [-ll] [-O] [-v]

config -b [-d] [-l]

config -s [-v]
```

Alias:	`conf`
`-a`	Include names of actors on each PE.
`-A`	Include application PEs.
`-b`	Display the boot information block (`typedef boot_info_t`).
`-C`	Include command PEs.
`-d`	Include an `od`-format dump of the boot information block.
`-l`	Generates a long report. See directive forms for details.
`-ll`	Generates a longer report. See directive forms for details.
`-O`	Include OS PEs.
`-p`	Display the PE configuration.
`-s`	Display the system configuration, a reconstruction of the parsed param file.
`-v`	Include entry address.

If neither `-b` nor `-p` nor `-s` option is specified, the default is `-p`.

If options are specified that have no meaning for a display, those options are silently ignored. For example, if `config -pd` is specified, `-d` is ignored.

Directive forms are:

`config -p [-a] [-A] [-C] [-l] [-ll] [-O] [-v]`

Display the PE configuration.

`-l`	Shows details of each PE.
`-ll`	Include bogus PE types.

If neither `-A` nor `-C` nor `-O` options are specified, the default is `-ACO`.

The `-a` and `-l` options are mutually exclusive. If both are specified, `-l` takes precedence.

If the `-v` option is specified, `-l` is implied.

PEs that are not started (i.e., the count of started actors is zero) are shown only in the detailed (`-l` option) display.

`config -b [-d] [-l]`

Display the boot information block (`typedef boot_info_t`).

`-l`	Include the values of `crashmk` internal variables that are based on information from the boot information block.

`config -s [-v]`

Display the system configuration.

⚠ **Caution:** The output of this display is lengthy, generally in the neighborhood of thousands of lines. Redirecting it to a file and viewing the file with an editor is recommended.

2.3.2.16 `console`

Display console buffers

```
console [-C] [-e] [-K] [pe# ...]
```

Alias:	`cons`
`-C`	Include the console (tty) output buffer.
`-e`	Display kernel output buffers on all sites.
`-K`	Include the kernel output buffer.
pe#	Hexadecimal integer. Specifies the number of the PE whose kernel buffer is to be displayed. The default is the current PE.

If no options or arguments are specified, the default is to display the kernel output buffer of the current PE (`-K` option).

If `-C` is specified and `-K` is not, only the tty buffer is displayed.

The `-e` option is ignored when *pe#*s are specified.

The `-e` option or *pe#* is ignored if kernel buffer display, `-K`, is not specified.

2.3.2.17 `ctx`

Display thread `KnThreadCtx` structure values.

```
ctx [-n] [-v] address
```

Aliases:	NONE
`-n`	Include the name of the nearest symbol or pool for values that appear to be addresses.
`-v`	Include mapping of register numbers shown in `totalview(8)` disassembly of kernel code to `KnThreadCtx` registers.
address	Address of a `KnThreadCtx` structure.

2.3.2.18 `debug`

Enables/disables debug mode.

```
debug [{ on || off || num }]
```

Aliases:	NONE
`on`	Enables debug mode.
`off`	Disables debug mode.
num	A number: `0` disables, non-zero enables.

If no argument is specified, the debug mode state is toggled.

When debug mode is enabled information about `crashmk` internal processing is displayed. This information is generally of no value to other than `crashmk` developers.

2.3.2.19 `default`

Sets or displays defaults.

```
default

default act[or] [actor]

default addr[mode] [addr_mode]

default boot[pe] [pe#]

default mode [mode]

{ default mem[ory] || memsiz } [maxmem [pe_range ...] ]

{ default numpe[s] || numpe[s] } [num]

{ default pe || gope } pe#
```

Aliases: `def`, `set`

`default`

Display current actor, machine, and mode settings.

`default act[or]` [*actor*]

Set default actor to *actor*, if specified. Otherwise, clear default actor setting.

`default addr[mode]` [*addr_mode*]

Set the addressing mode to *addr_mode*. Valid *addr_mode* settings are `byte`, and `word`. Addresses specified in arguments are assumed to be in bytes when in byte mode, and in words when in word mode. If *addr_mode* is omitted, the current setting is displayed.

`default boot[pe]` [*pe#*]

If *pe#*, a hexadecimal value, is specified, the boot PE number is set to *pe#*. Otherwise, display the boot PE number.

See Section 2.3.1.3, page 12.

`default mode` [*mode*]

Set default display mode to *mode*, if specified. The default mode is `hch`.

`default mem[ory]` [*maxmem* [*pe_range* ...]]

If *maxmem* is specified, set the memory size to *maxmem* for the specified PEs. The default *pe_range* is the current PE. If *maxmem* is omitted, the memory size of the current PE is displayed.

The value of *maxmem* is assumed to be in megabytes and is limited to 64, 128, 256, 512, 1024, or 2048. If `?` is specified, the user is prompted to select from these choices.

> **Note:** The memory size value is used as a limit for reads and writes done by `crashmk`(8). A M-option channel range error can occur when `crashmk` attempts to read or write beyond the limits of memory. Under normal circumstances the memory size is read from the `KnUmkContext` structure on the current PE, and *maxmem* need not be specified. This directive should be used only if the `KnUmkContext` structure has not been initialized or is corrupt.

`memsiz` is an alias for `default memory`.

`default numpe[s]` [*num*]

Set the number of PEs to *num*, if specified. The default is to display the number of PEs. *num* is an integer. The highest PE number will be *num* - 1, in hexadecimal.

See Section 2.2.4, page 8.

`numpe[s]` is an alias for `default numpe[s]`.

`default pe` *pe#*

Switch to PE *pe#*, where *pe#* is a hexadecimal number.

`gope` is an alias for `default pe`.

`bootpe` is an alias for `default pe` *boot_pe#*, where *boot_pe#* is the number of the boot PE.

2.3.2.20 `dfsstat`

Display DFS statistics.

```
dfsstat [-c] [-g] [-m] [-p] [-r]
```

Aliases:	NONE
`-c`	Display `CM` statistics.
`-g`	Display output in a form suitable for a search using `grep`(1).
`-m`	Display RPC memory usage.
`-p`	Display `PX` statistics.
`-r`	Display RPC statistics.

DFS statistics are kept in the root file server only when DFS is configured.

2.3.2.21 `dmptab`

Display the PE dump table pointer (`struct DmpTabPtr`) and its associated dump table entries (`typedef DmpTab`).

```
dmptab
```

Aliases:	NONE

2.3.2.22 `dnlc`

Display directory name lookup cache (DNLC) information.

```
dnlc [-e] [-n] [-p] [-s] [-v] [-w] [ [pe_range:][slot_range] ...]
```

Aliases:	NONE
`-e`	Display entries on all PEs where DNLC entries exist.
`-n`	Display entries following the `lru_next` chain, in order of least to most recently used.
`-p`	Display entries following the `lru_prev` chain, in order of most to least recently used.
`-s`	Include DNLC usage statistics.
`-v`	Include unused entries (that is, entries that have no parent vnode link). The default is to omit these entries.
`-w`	Include a list of PEs where a file server is loaded.

pe_range	A range of PE numbers. Numbers are assumed to be hexadecimal, and the trailing colon (`:`) is required. The default is the current PE if it has a file server. Otherwise, the default is the root file server's PE. Numbers of PEs that do not have file servers are silently ignored.
slot_range	A range of DNLC entry (slot) numbers. Unused entries within the range are displayed only if the `-v` option is specified. The default is all entries in entry number order. *slot_range* is ignored when the `-n` or `-p` option is specified.

See Section 2.3.1.5, page 14, for details of the range format.

Directory name lookup cache is found only on PEs with a file server.

Arguments are ignored when the `-e` option is specified.

The `-n` and `-p` options are mutually exclusive. If both are specified, the `-p` option takes precedence.

If the `-s` option is specified without the `-e`, `-n`, or `-p` options, or *slot_range* arguments, only DNLC usage statistics are displayed. Statistics for all file servers can be displayed by using `dnlc -s all:`.

If the `-w` option is specified without other options or arguments, only file server location information is displayed.

The construct `ncache[`*slot#*`]` can be used as an address argument. For example, `od ncache[21] 4` will display the first 4 words of DNLC entry 21.

2.3.2.23 `ds`

Display the name of the symbol whose value is closest to, but not greater than, the specified address, and the offset from that symbol.

```
ds [-a] address ...
```

Aliases:	NONE
`-a`	Shows all occurrences of *address*, if *address* is an unqualified symbol.

address	An address or symbol. See Section 2.3.2.4, page 17.

If the symbol is within an actor, the symbol display includes the actor's name. The display is of the form *actor.symbol* if the actor is loaded, or *actor!symbol* if the actor is not loaded.

The `ds` directive knows about certain internal structures, such as file system tables and pools. If *address* is within one of these structures, *symbol* is followed by an equal sign, then by the name of the structure and offset into that structure.

```
> ds 0x3fa6824
      GPM.edata + 0x30923ec = &Actor[2] + 0xc
```

Lists of known tables or known pools are included in the respective on-line helps, `help ds` or `help pool`.

See also the `ts` directive, Section 2.3.2.87, page 74.

2.3.2.24 errec

Display error data.

```
errec [-a] [-A] [-C] [-e] [-I] [-l] [-ll] [-M] [-r] [-t] [-v] [-x]
   [server ...]
```

Aliases:	`erec`, `erlog`
`-a`	Display all data; equivalent to `-eCIMRS`.
`-A`	Display all interrupt data; equivalent to `-CIMRS`.
`-C`	Include C–option error data.
`-e`	Include server's most recent error log entry.
`-I`	Include I–option error data.
`-l`	Include `od`-format dump of error log data.
`-ll`	Include full dump of error log record data when its size exceeds the sanity check value of 300 bytes.
`-M`	Include M-option error data.
`-R`	Include R-option error data.
`-S`	Include error interrupt data.
`-t`	Include time stamp in alternate (`-x` option) display.

`-v`	Verbose. Include the names of servers that have no error log entries. Unconditionally include dumps of `C`-, `I`-, `M`-, `R`-option error data and error interrupt data areas
`-x`	Display hardware option error data in a 1–line format. Option types (C, I, M, or R) are not shown in this display, so only one type should be displayed at a time. This format is intended for use by hardware engineers. It is designed so that the output can be passed directly into hardware analysis tools.
server	Names of servers for which error log entries are to be displayed.

If no error data type is specified, the default is all (`-a`).

If `-e` is specified or implied, and *server* is not specified, the default is all servers on the current PE.

If `-x` is specified, the `-e` and `-S` options are ignored.

2.3.2.25 `eslice`

Display `eslice` structures.

```
eslice [-e] [-l] [-v] [-w] [ [pe_range:][device] ...]
```

Alias:	`esl`
`-e`	Display all active entries on PEs where eslices exist.
`-l`	Include `eslice` details.
`-v`	Include `eslice` profile information.
`-w`	Include a list of PEs where a disk server is loaded.
pe_range	A range of PE numbers. Numbers are assumed to be hexadecimal, and the trailing colon (`:`) is required. The default is the current PE if a disk server is loaded. Otherwise, only a list of PEs where disk servers are loaded is displayed. See Section 2.3.1.5, for details of the range format.
device	Eslice device identifier. Can be one of the following forms:

address	Address of an `eslice` structure. Data at the specified address will be formatted as if it were an `eslice` structure. No validation is done.

major[`,` *minor*]	Device major number, minor number pair. Where *major* is the device major number, which can be an integer or a node name (for example, `dev_mdd`, or simply `mdd`). And *minor* is the device minor number, which must be an integer. If *minor* is omitted, all active eslices of the specified *major* device are displayed.

Eslices are found only on PEs with a disk server.

If the `-w` option is specified without arguments, only disk server information is displayed. Otherwise, if no arguments are specified, the default is to display all active eslices on the current PE.

Arguments are ignored when the `-e` option is specified.

Examples:

`eslice`	Display all active eslice entries.
`eslice dev_mdd`	Display all active eslices for mirrored devices.
`eslice dev_ldd/1`	Display the eslices for logical device minor 1.
`eslice 34/2`	Display the eslices for device major 34, minor 2.
`eslice 0x5d9f0c0`	Display the data at `0x5d9f0c0` assuming it to be an `eslice` structure.

2.3.2.26 `file`

Display file table entries.

`file [-a] [-e] [-l] [-w] [ [`*pe_range*`:][`*slot_range*`] ...]`

Alias:	`f`, `files`
`-a`	Include entries having a reference count of zero. This option is assumed when arguments are specified.
`-e`	Display all in use entries on all PEs where a file table exists.
`-l`	Include details of the associated `nc1inode` or `rnode.`
`-w`	Include a list of PEs where a file server is loaded.
pe_range	A range of PE numbers. Numbers are assumed to be hexadecimal, and the trailing colon (`:`) is required. The default is

the current PE if it has a file server. Otherwise, the default is the root file server's PE. Numbers of PEs that do not have file servers are silently ignored.

slot_range	A range of file entry (slot) numbers.

See Section 2.3.1.5, page 14, for details of the range format.

File tables are found only on PEs with a file server.

If the `-w` option is specified without arguments, only file server information is displayed. Otherwise, if no arguments are specified, the default is to display all in use entries on the default PE.

Arguments are ignored when the `-e` option is specified.

2.3.2.27 `find`

Search memory for occurrences of a pattern.

```
find [-a] [-d] [-k] [-m mask] [-n] pattern address [range]
```

Aliases:	`scan`, `search`
`-a`	Display all occurrences found. By default, only the first occurrence is displayed.
`-d`	Search in descending address order (that is, from *address* to *address* - *range*). By default, the search is in ascending address order (that is, from *address* to *address* + *range*).
`-k`	*pattern* is a `KSEG` address. The `KSEG` prefix, `0xfffffc0000000000`, is ORed with *pattern* and the result is used in the search.
`-m` *mask*	A 64–bit numeric value which is ANDed with the content of a search location before it is compared with *pattern*. If *mask* has less than 64 bits, it is right-justified in a word and zero filled to the left. The default mask depends on the pattern type. If the pattern is a numeric value, the default mask is 64 bits of 1's. If the pattern is a string, the default mask has `0xff` in the bytes corresponding to the string characters and zeros elsewhere. For example, the default mask for the pattern `"KMLW"` is `0xffffffff00000000`.
`-n`	Include the name of the symbol or pool nearest to the address of the match.

pattern	A 64–bit (word) pattern. It may be a numeric value, or a character string. If *pattern* is a string, it must be enclosed in quotes (for example, `"AbCd"`). Numeric values having less than 64 bits are right-justified in a word, and zero filled to the left. It if has more than 64 bits, the left-most bits are truncated. Strings having less than 64 bits are left-justified in a word, and zero filled to the right. if it has more than 64 bits, the right-most bits are truncated.
address	The **byte** address at which the search is to begin. The address must be on a word boundary. If not, it is rounded to the next lower word boundary for ascending searches, or the next higher word boundary for descending searches. If the keyword `+` is specified, the search continues from the word following the most recent match.
range	The number of **words** to search. By default, *range* is 1024 words. The prefix `b:` can be used to indicate a byte value, which will be converted to words. If the keyword `all` is specified, then *range* is from *start* to the end of memory.

Note: When the system image is a dump there can be gaps between the segments of memory that were dumped. These gaps are silently skipped over when included in the search range.

2.3.2.28 `flow`

Controls FlowTrace operation.

```
flow [{ -e || -r || -s || pe_range }]
```

Aliases:	NONE
`-e`	Ends (stops) FlowTrace data gathering.
`-r`	Resets FlowTrace counters.
`-s`	Starts FlowTrace data gathering.
pe_range	A range of PE numbers for which FlowTrace status is to be displayed. The default is the current PE. See Section 2.3.1.5, page 14, for details of the range format.

Only one option may be specified at a time, and arguments may not be specified with options.

The options control the gathering of kernel FlowTrace data, and are effective only when the *system_image* is a of a live system. They are set on the boot PE, and are propagated to all other PEs by the operating system.

Permission to write to the *system_image* is **required**.

FlowTrace data is displayed using the `kftrace` directive, Section 2.3.2.33, page 38.

Note: The Flow Tracing capability is present in specially built versions of the UNICOS/mk operating system only. It is not present in the generally released versions.

2.3.2.29 `gtb`

Decode Global Translation Buffer (GTB) entry.

```
gtb [-d] [value ...]
```

Aliases:	NONE
`-d`	Display the actual mask and page frame number (pfn) fields.
value	32–bit GTB translation array entry. *Value* can be a 64–bit word, containing two entries. In this case, both entries are decoded.

2.3.2.30 `help`

Display directive information.

```
help [ [-e] [-E] directive ...]
```

Aliases:	`h`, `?`
`-e`	Display information about *directive* output.
`-E`	Display information about *directive* syntax and output.
directive	Directive about which information is wanted.

If *directive* is not specified, a summary of all directives is displayed.

If both `-e` and `-E` options are specified, `-E` takes precedence.

Note: Some directives have long descriptions and/or explanations. Piping `help`'s output to your favorite pager is suggested. For example, `help -e proc | more`.

2.3.2.31 `info`

Display `info` server cache.

```
info [-l] [-s pe#] [slot# ...]
```

Aliases:	NONE
`-l`	Include dump of cache buffer.
`-s` *pe#*	PE number. The number is assumed to be hexadecimal. The default is the boot PE.
slot#	`info` server cache entry (slot) number. If omitted, all in use entries are displayed.

2.3.2.32 `job`

Display Global Job Table information.

```
job [-c] [-J] [-l] [job_id ...]
```

Aliases:	NONE
`-c`	Include file system I/O related system call statistics. This information is included when the `-l` option is specified.
`-J`	Interpret *job_id* as initiator's job id number (PJID).
`-l`	Include additional job information.
job_id	A range of job ids.

- When the `-J` option is not specified, the default is all entries.
- When the `-J` option is specified, *job_id* is required. Job table entries of the initiating and initiated jobs are displayed.

See Section 2.3.1.5, page 14, for details of the range format.

2.3.2.33 `kftrace`

Display kernel FlowTrace information.

```
kftrace [-a] [-c] [-f] [-k] [{ -m || -M }] [-n] [-s] [-t] [-U]
   [-x num]
```

Alias:	`kft`
`-a`	Inverse sort by average time in function.
`-c`	Reports times in microseconds. Otherwise, times are in clock periods.
`-f`	Inverse sort by frequency called.
`-k`	Keep this trace data image; allowing the same data to be displayed in different ways by subsequent `kftrace` directives. By default, data is re-read from kernel tables. This option has meaning only when processing a live system.
`-m`	Include the address of the named routine, allowing differentiation of multiple instances of the same name.
`-M`	Include the name of the server within which the function is found.
`-n`	Sorts by function name.
`-s`	Summarize extended trace information, omitting calling function information.
`-t`	Inverse sort by total time in function.
`-U`	Unconditional, ignore certain questionable entries.
`-x` *num*	Exclude functions called *num* times or less.

If no sort options are specified, the default is `-t`.

If more than one sort option is specified, a separate report is produced for each option.

The `-m` and `-M` options are mutually exclusive. If both are specified, `-m` takes precedence.

FlowTrace operation is controlled using the `flow` directive, Section 2.3.2.28, page 35.

Note: The Flow Tracing capability is present in specially built versions of the UNICOS/mk operating system only. It is not present in the generally released versions.

2.3.2.34 kmem

Display dynamic kernel memory control information.

```
kmem [-b] [-d] [-f] [-l] [-ll] [-lll] [-llll] [-L] [-s] [-v] [count]
```

Aliases:	NONE
`-b`	Display call chain information grouped by common calling sequences only. Control information is excluded
`-d`	Display list of corrupted kmem allocations.
`-f`	Check for valid patterning in free kmem. When specified, the `-l` option is assumed. **Note:** This process can be time consuming, as each word of free kmem is examined.
`-l`	Display call chain information grouped by common calling sequences.
`-ll`	Display call chain information separately for each allocated block.
`-lll`	Includes raw page sub-allocations. **Note:** The output of this and the following option's display is lengthy. Redirecting it to a file and viewing with an editor is recommended.
`-llll`	Include traceback information.
`-L`	Equivalent to `-llll`.
`-s`	Display a summary of the usage of each block. This display may be useful in identifying fragmentation.
`-v`	Verbose. Include calling function' argument list, when available. By default, only function names are shown, and the display is truncated at 80 characters.

count	Number of lines of call chain information entries to be displayed. The keyword `all` can be used to specify all entries. The default is all entries.

If *count* is specified without options, the `-l` option is assumed.

If `-b` is specified without other options, call chain information is grouped by common calling sequences, and no control information is displayed.

2.3.2.35 `kstack`

Display kernel virtual stack information.

```
kstack { [-a] [-s] | [-x_range ...] }
```

Aliases:	`kstk`, `kvs`
`-a`	Display all kernel virtual stacks.
`-s`	Display historical stack size summary.
ix_range	A range of kstack indices. A kstack index is identical to the slot number of its associated thread. See Section 2.3.1.5, page 14, for details of the range format.

There is no default display.

2.3.2.36 `lkup`

Display symbols that match a pattern.

```
lkup [-i] str ...
```

Alias:	`grep`, `nm`, `syms`
`-i`	Do case-insensitive search.
str	String or pattern to match.

2.3.2.37 `loadtable`

Display boot actor (server)load table for the current PE.

```
loadtable [-l]
```

Alias:	`lt`
`-l`	Display the entire load table. By default, only entries for servers that are loaded (i.e., configured) are displayed.

This table is used by the microkernel to load the servers. The physical address and size of each server is displayed.

2.3.2.38 `logout`

Display the logout structure. Internal registers are copied to the logout structure when a PE halts.

```
logout [-a] [-b] [-c] [-d] [-e] [-i] [-m] [-u] [-v] [-C] [-E] [-F]
  [-I] [-M] [-R]
```

Alias:	`log`
`-a`	Display all information; equivalent to `-bcdeimuCIMR`.
`-b`	Include BESU (barrier/eureka synchronization) information.
`-c`	Include EV5 CBOX (cache control) registers.
`-d`	Include EV5 DTB/ITB (data/instruction translation buffer) contents.
`-e`	Include kernel E-register set.
`-i`	Include EV5 IPRs (internal processor registers).
`-m`	Include MMRs (memory mapped registers).
`-u`	Include user E-register set.
`-v`	Verbose — gives descriptive interpretation of registers.
`-C`	Include `C`-option (control) registers.
`-E`	Explains (decode) fields in registers.
`-F`	Force display when the value of the logout flag indicates the specified section of the logout area has not been completely filled.
`-I`	Include `I`-option (I/O) registers.
`-M`	Include `M`-option (memory) registers.

`-R`	Include `R`-option (network router) registers.

2.3.2.39 `map`

Display system `map` structures.

```
map [-l] [-ll] [-m] [{ name | address } ...]
```

Aliases:	NONE
`-l`	Include list of allocated and free areas.
`-ll`	Include list of allocated and free areas, and dump of map words.
`-m`	Include dump of map words.
name	Map structure name. A list of known map structures is included in the on-line help, `help map`.
address	Map structure address. Data at the specified address will be formatted as if it were a `map` structure.

If no arguments are specified, all known map structures associated with servers on the current PE are displayed.

2.3.2.40 `mem`

Display memory size information from the `KnUmkContext` structure on the current PE.

```
mem
```

Aliases:	NONE

2.3.2.41 `mdw`

Display dump header memory descriptor words.

```
mdw
```

Aliases:	`mdws`, `msd`, `msds`

2.3.2.42 `mmu`

Display the Memory Management Unit (MMU) translation data.

```
mmu [-a] [-r] [-s segment] address
```

Aliases:	NONE
`-a`	Include Page Table Entry (PTE) details for all segments.
`-r`	Display Page Table Entry (PTE) data in raw form; information is not decoded.
`-s` *segment*	Display Page Table Entry (PTE) details for *segment*.
address	Address of a `mmu_t` (MMU structure).

2.3.2.43 `mount`

Display mount table entries.

```
mount [-a] [-e] [-l] [-ll] [-r] [-w] [ [pe_range:][slot_range] ...]
```

Aliases:	`mnt`, `m`
`-a`	Include free entries. This option is assumed when arguments are specified.
`-e`	Display all in use entries on all PEs where a mount table exists.
`-l`	Include the `vfs` entry and `attribute` structure.
`-ll`	Include super-block and dynamic block information.
`-n`	Include NFS mount information structure details.
`-r`	Include remote mount information.
`-w`	Include a list of PEs where a file server is loaded.
pe_range	A range of PE numbers. Numbers are assumed to be hexadecimal, and the trailing colon (`:`) is required. The default is the current PE if it has a file server. Otherwise, the default is the root file server's PE. Numbers of PEs that do not have file servers are silently ignored.

slot_range	A range of mount entry (slot) numbers.

See Section 2.3.1.5, page 14, for details of the range format.

Mount tables are found only on PEs with a file server.

If the `-r` and/or `-w` options are specified without arguments, only remote mount and/or file server information is displayed. Otherwise, if no arguments are specified, the default is to display all in use entries on the default PE.

Arguments are ignored when the `-e` option is specified.

2.3.2.44 `msgq`

Display IPC message queues.

```
msgq [-h] [-l] [-t]
```

Alias:	`mq`
`-h`	Include an interrupt time histogram.
`-l`	Include a detailed display of the `MQ` class for each queue.
`-t`	Display interrupt time data only.

By default, the display shows a summary of all types of interrupts, a summary of message type interrupts, and the state and some of the counters for each message queue.

2.3.2.45 `nc1inode`

Display `nc1inode` table entries.

```
nc1inode [-a] [-e] [-l] [-ll] [-O] [-p] [-w] [ [pe_range:][slot_range] ...]
```

Aliases:	`in`, `inode`, `nc`
`-a`	Include unused entries. This option is assumed when arguments are specified.
`-e`	Display all in use entries on all PEs where a `nc1inode` table exists.
`-l`	Include details of the `nc1inode`, `vnode`, and `attribute` structures.

`-ll`	Include `vnode` operations trace and SFS lock information.
`-O`	Display orphan inodes only. That is, inodes where the type (`v_type`) is not directory, pipe, or link, the reference count (`v_count`) is zero, the link count (`v_nlink`) is non-zero, and no file table entry points to it.
`-p`	Include PID of the last process to lock the inode. This field replaces the file size field in the summary line.
`-w`	Include a list of PEs where a file server is loaded.
pe_range	A range of PE numbers. Numbers are assumed to be hexadecimal, and the trailing colon (`:`) is required. The default is the current PE if it has a file server. Otherwise, the default is the root file server's PE. Numbers of PEs that do not have file servers are silently ignored.
slot_range	A range of `nc1inode` entry (slot) numbers.

See Section 2.3.1.5, page 14, for details of the range format.

`nc1inode` tables are found only on PEs with a file server.

If the `-w` option is specified without arguments, only file server information is displayed. Otherwise, if no arguments are specified, the default is to display all in use entries on the default PE.

Arguments are ignored when the `-e` option is specified.

2.3.2.46 `nfsmi`

Display NFS mount information structures.

`nfsmi` [*pe_range*`:`]*address* ...

Aliases:	NONE
pe_range	A range of PE numbers. Numbers are assumed to be hexadecimal, and the trailing colon (`:`) is required. The default is the current PE if it has a file server. Otherwise, the default is the root file server's PE. Numbers of PEs that do not have file servers are silently ignored. See Section 2.3.1.5, page 14, for details of the range format.

address	Structure address. Data at the specified address will be formatted as if it were a NFS mount structure.

NFS mount information structures are found only on PEs with a file server.

No validation of the address is done. Data at the specified location will be formatted as if it were a NFS mount information structure.

2.3.2.47 `od`

Dumps data.

```
od [-m mode] [-n] [-o] [-O] start [count [mode]]
```

Aliases:	NONE
`-m` *mode*	Display mode. The default is the current display mode, as set by the `default mode` directive. See Section 2.3.1.3, page 12, for valid *mode* values.
`-n`	Include the name of the nearest symbol or pool for data that appear to be addresses. This option is ignored when the data display is not a full word.
`-o`	Enables display of relative offset. The offset is displayed along with the address, and is relative to the starting address specified when the option is specified. It remains in effect for subsequent `od` directives until either the `-O` or another `-o` option is specified.
`-O`	Disables display of relative offset.
start	Starting address. Can be either of the following forms:

address	Address value.
`+`	Continue from the last address displayed plus 1.

count	Numeric value that specifies the number of data elements to dump. The size of a data element is byte, character, parcel or word, depending on the *format* specification. Default: `1`
mode	Display mode. If not specified, the default is the current `default mode` setting. If *mode* is specified by both the `-m` option and this positional argument, the `-m` option specification takes precedence.

See Section 2.3.1.3, page 12, for valid *mode* values.

The `-o` and `-O` options are mutually exclusive. If both are specified, `-o` takes precedence.

For byte address mode:

- *Address* is expected to be word-aligned, and is rounded down to the next lower word boundary when it is not.
- A numeric *address* followed by a `W` indicates a word address.

For word address mode:

- A numeric *address* followed by a `p` or `B` indicates a parcel or byte address, respectively.
- A numeric *address* followed by an `a`, `b`, `c`, or `d` indicates respectively the first, second, third, or fourth parcel within a word.
- The suffixes `a`, `b`, `c`, `d`, and `B` are assumed to be part of a hexadecimal *address*.

2.3.2.48 onpe

Executes a directive on specified PEs.

```
onpe [-A] [-C] [-d] [-O] [-s] [-U] [-V] pe_def do directive
```

Alias:	`onpes`
`-A`	Executes *directive* on all application PEs.
`-C`	Executes *directive* on all command PEs.
`-d`	Set default actor to *server* before executing *directive*. This option is effective only when specified in conjunction with the `-s` option.
`-O`	Executes *directive* on all OS PEs.
`-s`	Interpret *pe_def* as a server name.
`-U`	Unconditional. Continue if an error status is returned from *directive*. By default no further attempts to execute *directive* are made after an error status is returned.
`-V`	Non-verbose mode. Suppresses `On PE` *pe#*`:` messages.
pe_def	Defines the PEs where *directive* is to be executed.

- The default form is *pe_range*. A range of PEs where *directive* is to be executed. See Section 2.3.1.5, page 14 , for details of the range format.
- When the `-s` option is specified, the form is *server*—the name of one or more servers.

directive	Any valid `crashmk` directive.

`onsrvpe` is an alias for `onpe -s`.

pe_def may be omitted when the PE range options `-A`, `-C`, or `-O` are specified.

When PEs are specified by both options and arguments, PEs specified by options are visited first, then PEs specified by arguments. A PE may be visited more than once if the PEs specified by the options and the PEs specified by arguments overlap, and, similarly, when the `-s` option is specified and the PEs where servers are found overlap.

Examples:

```
onpe -O 100 110 7f-70 do od kmem_allocs_failed
```

Executes `od kmem_allocs_failed` on all OS PEs, and on PEs 0x100, 0x110, and 0x7f through 0x70, inclusive.

```
onpe -s packet disk do erec
```

Executes `erec` first on PEs where there are packet servers, then on PEs where there are disk servers. Note that PEs having both packet and disk servers will be visited twice.

2.3.2.49 `paltrace`

Display MKPAL trace entries.

`paltrace [-1] [-A] [-n] [-P` *address*`] [-r] [-T] [-v] [+] [`*count*`]`

Alias:	`pt`
`-1`	Display all entries (*count* specification is ignored).
`-A`	Display all entries (*count* specification is ignored).
`-n`	Include the name of the nearest symbol or pool for trace interrupt addresses.

`-P` *address*	Address of initial MKPAL trace entry. If not specified, `crashmk` attempts to determine the address. This option is ignored when `+` is specified.
`-r`	Display entries without decoding the information (raw mode).
`-T`	Display the time stamp in raw form. By default, the time stamp is displayed in an absolute time-of-day from with no adjustment for current time.
`-v`	Include the entry's address.
`+`	Continue trace from next oldest entry. This specification is ignored when the *system_image* is of a live system.
count	Number of entries to display (default: 24). The following keyword is also allowed: `all` Display all entries.

2.3.2.50 `pddtab`

Display disk `pdd` table entries.

```
pddtab [-a] [-e] [-l] [-ll] [-v] [-w] [ [pe_range:][slot_range] ...]
```

Alias:	`pdd`
`-a`	Include unused entries. This option is assumed when arguments are specified.
`-e`	Display all in use entries on all PEs where a `pdd` table exists.
`-l`	Generates a long report for `pdd_tab` structures.
`-ll`	Generates a very long report for `pdd_tab` structures, including a trace of active and outstanding buffers.
`-v`	Include `pdd_tab` profile information.
`-w`	Include a lit of PEs where a disk server is loaded.
pe_range	A range of PE numbers. Numbers are assumed to be hexadecimal, and the trailing colon (`:`) is required. The default is the current PE if a disk server is loaded. Otherwise, only a list of PEs where disk servers are loaded is displayed.

slot_range	A range of `pdd` table entry (slot) numbers.

See Section 2.3.1.5, page 14, for details of the range format.

Pdd tables are found only on PEs with a disk server.

If the `-w` option is specified without arguments, only disk server information is displayed. Otherwise, if no arguments are specified, the default is to display all active entries on the current PE.

Arguments are ignored when the `-e` option is specified.

2.3.2.51 `pemap`

Display PE allocator's map entries.

```
pemap [-g gid] [-l] [-ll] [-p num] [-s] [-u uid] [first [number]]
```

Alias:	`pem`
`-g` *gid*	Display PEs having *gid* in the authorization list.
`-l`	Display application queue pointers.
`-ll`	Include display of pointers to all structures read.
`-p` *pid*	Display PEs allowing *num*-PE applications.
`-s`	Display plane shape information.
`-u` *uid*	Display PEs having *uid* in the authorization list.
first	First PE's logical number. `.` means most recent *first*. `+` means most recent *first* + 1.
number	Number of PEs to display. `$` means all PEs from *first* to the highest logical PE. If not specified, the most recent previously entered value is used. The default is all PEs.

If `-s` is specified alone, only the plane shape is displayed.

For `-g` and `-u`, only PEs with authorization lists are selected.

2.3.2.52 pequeue

Display PE allocator's application queue entries.

```
pequeue [-a apid] [-B] [-g gid] [-l] [-ll] [-q] [-r] [-s] [-u uid]
    [address]
```

Alias:	`peq`
`-a` *apid*	Display the entry for application *apid*.
`-B`	Display detailed barrier tree setup information.
`-g` *gid*	Display entries with group *gid*.
`-l`	Display barrier setup array address.
`-ll`	Include display of pointers to all structures read.
`-q`	Display queued applications.
`-r`	Display running applications.
`-s`	Display queue statistics.
`-u` *uid*	Display entries having owner uid.
address	Address of queue entry. Queue entry addresses are displayed using `pemap -l`.

If neither `-q` nor `-r` are specified, the default is all entries.

All `-a`, `-g`, and `-u` selections must be satisfied to display an entry.

If `-s` is specified alone, only queue statistics are displayed.

2.3.2.53 pmfsst

Display Process Manager's (PM server) file system I/O system call counters.

```
pmfsst [-t] [pe_range ...]
```

Aliases:	`pmfs`, `pmstats`
`-t`	Display totals of counters and times over the range of specified PEs. The default is a separate display for each PE.

pe_range	A range of PE numbers. The default is the current PE.

See Section 2.3.1.5, page 14, for details of the range format.

2.3.2.54 `poke`

Writes to memory (live system image only).

```
poke [-U] address value [count]
```

Aliases:	NONE
`-U`	Unconditional: **do not** ask for confirmation before writing.
address	Address of the first memory location to be written to. *address* **must** be word-aligned.
value	The **64-bit** value to be written. If less than 64 bits, it is zero-filled to the left.
count	The number of **words** to write. If not specified, the default is 1.

`poke` replaces the words from *address* through *address* + *count* - 1 with *value*.

There are two possible interfaces between `crashmk` and a live system image `/dev/pe` and the `skdb` server. `/dev/pe` is available when `crashmk` is running under the UNICOS/mk operating system. The `skdb` server is used when `crashmk` is running on a remote host. The conditions under which writing is allowed depend on the interface being used:

- For `/dev/pe`, you must have write-access to the character special device `/dev/pe`.
- For the `skdb` server, you must be a member of one the groups `root`, `sys`, `os`, or `craysrc` on the host where `crashmk` is running.

2.3.2.55 `pool`

Display system pool structures.

```
pool [-l] [-ll] [name ...]
```

Alias:	`pools`
`-l`	Include the free list start and all pool elements that have data.

`-ll`	Include all pool elements.
name	Name of pool structure. If omitted, all known pool structures are displayed. The directive `help pool` displays the list of known pools.

2.3.2.56 pput

Validates the physical page usage table (`PPut`) and display page usage table information.

```
pput [-l] [-p paddr] [-r] [-U] [entry ...]

pput -f [-b] [-p paddr] [-r] [-U]

pput -h haddr [-b] [-p paddr] [-r] [-U]
```

Alias:	`pages`
`-b`	Backward: follows previous links when displaying a linked page list. The default is to follow the next link.
`-f`	Display information about the 64KB and 512KB free lists.
`-h` *haddr*	Address of the head of a linked page list (`PhysPageList`). The header and all pages on the list are displayed.
`-l`	Include a one-line description of all page entries in the page usage table, in entry number order.
`-p` *paddr*	Address of the page usage table. The default is `PhysPage::PPut`.
`-r`	Raw mode: link and entry addresses are not converted to page usage table entry (slot) numbers.
`-U`	Unconditional: continues processing even if any standard linked list headers cannot be found, or if errors were found during validation of the page lists.
entry	Page usage table entry. Can be either an entry (slot) number, or an entry address. A one-line description of each specified page entry is displayed.

The `-f` and `-h` options are mutually exclusive.

Arguments are ignored when `-f` or `-h` options are specified.

The `-l` option is ignored when arguments are specified.

The integrity of the page usage table is validated as follows:

- For each page list header, if one link points to itself, both must (that is, the list is empty).
- If a list is not empty, both links must point to page usage table entries.
- Links in page usage table entries must point to another page usage table entry, or to the page list header.
- Each page usage table entry may appear on at most one list.
- Lists are checked to see if they if they point into themselves (that is, are endless).
- The total size of all pages on a list must equal the size kept in the page list header.

Directive forms are:

`pput [-p` *paddr*`] [-r] [-U]`

Display page usage summary.

`pput -l [-p` *paddr*`] [-r] [-U]`

Display a one-line description of all page entries in the page usage table, in entry number order.

`pput -f [-b] [-p` *paddr*`] [-r] [-U]`

Display free list summary.

`pput -fl [-b] [-p` *paddr*`] [-r] [-U]`

Display free list summary and a one-line description of all page entries on free lists.

`pput -h` *haddr* `[-b] [-p` *paddr*`] [-r] [-U]`

Display the list header and a one-line description of all page entries on the list.

2.3.2.57 prnode

Display `prnode` (`/proc rnode`) entries.

```
prnode [-a] [-e] [-l] [-w] [ [pe_range:][slot_range] ...]
```

Alias:	`prn`
`-a`	Include unused entries. This option is assumed when arguments are specified.
`-e`	Display all in use entries on all PEs where a `prnode` table exists.
`-l`	Generates a long report that includes the `vnode` and `attribute` structures.
`-w`	Include a list of PEs where a file server is loaded.
pe_range	A range of PE numbers. Numbers are assumed to be hexadecimal, and the trailing colon (`:`) is required. The default is the current PE if it has a file server. Otherwise, the default is the root file server's PE. Numbers of PEs that do not have file servers are silently ignored.
slot_range	A range of `prnode` entry (slot) numbers.

See Section 2.3.1.5, page 14, for details of the range format.

`prnode` tables are found only on PEs with a file server.

If the `-w` option is specified without arguments, only file server information is displayed. Otherwise, if no arguments are specified, the default is to display all in use entries on the default PE.

Arguments are ignored when the `-e` option is specified.

2.3.2.58 proc

Display process information.

```
proc [-a] [-A] [-c] [-d] [-e] [-f] [-g] [-j] [-k] [-l] [-ll] [-L] [-n]
   [-p] [-P] [-q] [-r] [-s] [-t] [-u] [-v] [-W] [proc_id ...]
```

Aliases:	`p`, `ps`
`-a`	Include actor information.
`-A`	Include `apTeam` (multi-PE application) information.
`-c`	Include file system I/O related system call statistics. This information is also included when the `-ll` option is specified.
`-d`	Interpret *proc_id* as the address of a `tPmProc` structure.
`-e`	Display processes running on all PEs.
`-f`	Include file descriptor (FD) information. This information is also included when the `-ll` option is specified.
`-g`	Display GPM-based process information. It the `-e` option is specified, all process known to the GPM are displayed. Otherwise, the GPM's process table is displayed.
`-j`	Interpret *proc_id* as a job id number (JID).
`-k`	Include stack traces for threads of specified processes.
`-l`	Include process details.
`-ll`	Include PM thread and file descriptor information. If the `-a` and/or `-t` options are specified, actor and/or thread details are included.
`-L`	Include process limits. If specified, `-l` is implied.
`-n`	Include the name of the nearest symbol or pool for stack arguments that appear to be addresses. This option is ignored if the `-k` option is not specified.
`-p`	Interpret *proc_id* as a process id number (PID).
`-P`	Interpret *proc_id* as a parent process id number (PPID). The parent process and its children are displayed.
`-q`	Include actor swap information. If specified, the `-a` option is implied, and actor details are included.
`-r`	Include threads on remote sites that either created, or were created by the process's thread. This option is ignored if the `-t` option is not specified.
`-s`	Include extended signal information. If specified, `-ll` is implied.

`-t`	Include thread information. If specified, `-a` is implied.
`-u`	Interpret *proc_id* as a user id number (UID).
`-v`	Include idle threads. This option is ignored if the `-t` option is not specified.
`-W`	Wide line: includes all default and optional information in the one-line summary (the line may exceed 80 characters).
proc_id	Process identifier.

- The default form is [*pe_range*`:`][*slot_range*].

pe_range	A range of PE numbers. Numbers are assumed to be hexadecimal, and the trailing colon (`:`) is required. The default is the current PE.
slot_range	A range of process entry (slot) numbers. The default is all entries.

See Section 2.3.1.5, page 14, for details of the range format.

- If the `-d` option is specified, the form is *padr*. Where *padr* is tha address of a `tPmProc` structure.
- If the `-j` option is specified, the form is *jid*. Where *jid* is a job id number.
- If the `-p` option is specified, the form is *pid*. Where *pid* is a process id number.
- If the `-P` option is specified, the form is *ppid*. Where *ppid* is a parent process id number.
- If the `-u` option is specified, the form is *uid*. Where *uid* is a numeric user id.

By default, a one-line summary is displayed for each active process on the current PE.

If the `-e` option is specified, *proc_id* specifications are ignored.

If the `-t` or `-a` options are specified along with the `-l` option, thread and/or actor details are included.

If the `-A` option is specified without the `-W` option, virtual PE# (VPE) and `apTeam` id (APID) replace job-id (JID) and system-call/signal information (WCHAN) in the one-line summary.

If the `-W` option is specified, session id (SID) is included in the one-line summary, and the `apTeam` fields (VPE and APID) are shown in addition to the job-id (JID) and system-call/signal information (WCHAN).

The `-a`, `-A`, `-k`, `-n`, `-r`, `-s`, and `-t` options are ignored when the `-g` option is specified without the `-e` option.

The `-d`, `-j`, `-u` and `-p` or `-P` options are mutually exclusive. If both `-p` and `-P` are specified, `-P` takes precedence.

The `-g` option is ignored when the `-d` option is specified.

2.3.2.59 `psched`

Display political scheduling information.

```
psched [-c] [-l]
```

Aliases:	NONE
`-c`	Format the `PmSched` check in array, used by asynchronous calls to `PolicyLocal`().
`-l`	Generates long report, including information about the individual nodes.

2.3.2.60 `pty`

Display `pty` table entries.

```
pty [-l] [-ll] [ [-a] entry ...]
```

Aliases:	NONE
`-a`	Specifies that *entry* arguments are addresses.
`-l`	Generates a long report that includes the `stty(1)` parameters.
`-ll`	Generates a longer report that includes the `pt_ioctl` information.

entry	A pty number, or if -a is specified, an address of a pty entry. If omitted, all opened entries are displayed.

2.3.2.61 pwho

Display physical PE numbers (PWHOs) of specified logical PEs.

```
pwho [-1] [-A] [-c] [-C] [-O] [-x x_range] [-y y_range] [-z z_range]
    [pe_range ...]

pwho -m [-c] [pe_range ...]

pwho -n [-c] [pe_range ...]
```

Alias:	pwhos
-1	Display one PE per line. By default four PEs are displayed per line.
-c	Get PWHOs using config server information (i.e., param.conf). By default, the in-memory MK_LOGICAL2PHYSICAL table on the current PE is used.
-m	Display PEs on the same module (PEM) as the specified PEs.
-n	Display the immediate neighbors of specified PEs (i.e., PEs that are one hop away in either the X, Y, or Z plane).
-x *x_range*	Limit display to PWHOs having X-addresses within the range *x_range*.
-y *y_range*	Limit display to PWHOs having Y-addresses within the range *y_range*.
-z *z_range*	Limit display to PWHOs having Z-addresses within the range *z_range*.
-A	Include all application PEs.
-C	Include all command PEs.
-O	Include all operating system PEs.
pe_range	A range of logical PE numbers. The numbers are assumed to be hexadecimal. The default depends upon which options, if any, are specified. • If no options are specified, the default is the current PE.

- If only limiting options (`-x`, `-y`, and/or `-z`) are specified, the default is all PEs.
- If PE range options (`-A`, `-C`, and/or `-O`) are specified, there is **no** default.

Directive forms are:

`pwho [-1] [-A] [-c] [-C] [-O] [-x` *x_range*`] [-y` *y_range*`] [-z` *z_range*`]` [*pe_range* ...]

Display PWHOs of specified PEs.

`pwho -m [-c]` [*pe_range* ...]

Display PEs on the same module (PEM) as the specified PEs

`pwho -n [-c]` [*pe_range* ...]

Display the immediate neighbors of specified PEs.

See Section 2.3.1.5, page 14, for details of the range format.

The `-m` and `-n` options are mutually exclusive.

When the `-m` or `-n` options are specified all other options, except `-c`, are ignored.

No check is made for duplication between PEs specified by options and PEs specified by arguments. For example, the directive `pwho -A appe` will display PWHOs of all application PEs twice.

The effects of the limiting options (`-x`, `-y`, and/or `-z`) are additive. That is, a PWHO will be displayed only if it meets all the limits. PEs that are marked as down will not be displayed.

PEs on the same PEM as the specified PE are flagged with a `*` in the `-n` option display.

2.3.2.62 `quit`

Exit `crashmk`.

```
quit
```

Aliases:	`exit`, `q`

See Section 2.2.7, page 9, for a list of values returned when `crashmk` exits, along with their meaning.

2.3.2.63 `redirect`

Sends a copy of output to a file.

```
redirect [{ ? || [-s] [+] filename }]
```

Aliases:	>, >>, `redir`
`-s`	Silent mode: sends output to *filename* only. The default is to send output to both the file and to `stdout`.
+	Appends to specified file.
?	Display the name of current redirect file, if any.
filename	Name of file to receive output.

`redirect` without *filename* closes the current redirect file, if any. The >> alias causes the output to be appended to the specified file. It is equivalent to `redirect +`.

There must be a space between > or >> and *filename*.

Warning: The aliases > and >> **cannot** be the object of the `help` directive (that is, `help >`, or `help >>`). When > or >> is not the first argument in a directive, it is interpreted as specifying that output is to be written or appended to a file.

2.3.2.64 `reg`

Display registers. Register data is only available on a halted PE.

```
reg [-F] [reg_spec ...]
```

Alias:	`reg`
`-F`	Force display when the value of the logout halt flag indicates registers have not been saved.
reg_spec	Register number or register set.

The form is *reg_ty*[*reg_no*]. Where *reg_ty* specifies the register type, and *reg_no* specifies the register number. If *reg_no* is omitted, all *reg_ty* registers are displayed.

If no arguments are specified, all registers of all types are displayed.

Table 4. Register Types

Type (*reg_ty*)	Description
`i`, `ir`	Integer registers
`f`, `fp`	Floating point registers
`p`, `pt`	PAL temp registers
`ps`	PAL shadow registers

For example, the following directive displays integer registers 1 and 15, and all floating point registers.

```
reg i1 i15 fp
```

2.3.2.65 `reslist`

Display per-server resiliency (`resStruct`) structure.

`reslist [-l] [-n]` *address* [*pe#*]

Aliases:	NONE
`-l`	Include additional server dependent data.
`-n`	Include the name of the symbol or pool nearest to addresses that are displayed.
address	Address of a `resStruct` structure. Data at the specified address will be formatted as if it were a `resStruct` structure. No validation is done.
pe#	Display data for specified PE only. By default, data for all PEs defined by the `resStruct` structure is displayed.

Symbols which define `resStruct` structures are `resFileServer` and `resSksServer`.

2.3.2.66 rlimits

Display resource limits defaults.

```
rlimits [-v] [pe_range ...]
```

Aliases:	`rlimit`, `rlim`
`-v`	Verbose. Display limit description. By default, limit types are displayed.
pe_range	A range of logical PE numbers. The numbers are assumed to be hexadecimal. The default is the current PE. See Section 2.3.1.5, page 14, for details of the range format.

2.3.2.67 rmon

Display R-option (network router) performance monitor information.

```
rmon [-T]
```

Aliases:	NONE
`-T`	Display the time stamp in raw form.

Note: The amount of information displayed is substantial. Redirecting the output to a file is recommended.

2.3.2.68 rnode

Display `rnode` entries.

```
rnode [-a] [-e] [-l] -ll [-w] [ [pe_range:][entry] ...]
```

Alias:	`rn`
`-a`	Include unused entries. This option is assumed when arguments are specified.
`-e`	Display all in use entries on all PEs where a `rnode` table exists.
`-l`	Include locking and usage information.

`-ll`	Generates a long form report that includes the `vnode` and attribute structures.
`-w`	Include a list of PEs where a file server is loaded.
pe_range	A range of PE numbers. Numbers are assumed to be hexadecimal, and the trailing colon (`:`) is required. The default is the current PE if it has a file server. Otherwise, the default is the root file server's PE. Numbers of PEs that do not have file servers are silently ignored.
entry	Client handle table entry. Can be either of the following:

slot_range	A range of rnode entry (slot) numbers.
address	Address of a `rnode` entry. The address is ignored if it is not within the `rnode` table. Otherwise, the entry nearest to the address from below is displayed.

See Section 2.3.1.5, page 14, for details of the range format.

`rnode` tables are found only on PEs with a file server.

If the `-w` option is specified without arguments, only file server information is displayed. Otherwise, if no arguments are specified, the default is to display all in use entries on the default PE.

Arguments are ignored when the `-e` option is specified.

2.3.2.69 `route`

Display routes from the memory copy of `R_NET_LUT[543:0]` on the current PE.

```
route [-m mode] [-r] [-v] [pe_range ...]
```

Aliases: `routes`, `rte`

`-m` *mode* — Specifies the value to be used for `R_LUT_OVERRIDE_MODE`. Valid values are 0, 1, or 2.

The default is:

- The value from `R_LUT_OVERRIDE` that was saved at system initialization, or

- the value from R_LUT_OVERRIDE that was copied to the logout area if the PE has halted, or
- an estimated value based on the number of PEs.

-r	Include the raw image of the R_NET_LUT entries.
-v	Include the address of the R_NET_LUT entries.
pe_range	A range of PE numbers. If specified, routes from the current PE to these PEs are displayed. The default is all PEs. See Section 2.3.1.5, page 14, for details of the range format.

2.3.2.70 runq

Display run queue (runQueue) entries.

```
runq [-r]
```

Alias:	rq
-r	Dumps the run queue using an od-format.

By default, run queue header information and a list of queued threads is displayed. Sanity checking is done during this process to verify that links are pointing to threads.

2.3.2.71 rwlock

Display single writer / multiple reader lock information.

```
rwlock address ...
```

Aliases:	NONE
address	Address of a tRwLock structure. Data at the specified address will be formatted as if it were a tRwLock structure. Minimal validation is done.

2.3.2.72 sar

Display selected data in format similar to sar(1) utility.

```
sar [-b] [-e] [-n num] [-r rate] [-w] [pe_range ...]
```

Aliases:	NONE
`-b`	Display system buffer activity.
`-e`	Display data for all PEs where a file server is loaded.
`-n` *num*	Total number of samples to be collected. The default is 10. If zero is specified, there is no limit. This option is ignored if the system image is a dump.
`-r` *rate*	Display rate in seconds. The initial default is 30 seconds. When *rate* is specified, it becomes the default for subsequent uses of this directive. This option is ignored if the system image is a dump.
`-w`	Include a list of all PEs where a file server is loaded.
pe_range	A range of PE numbers. The number is assumed to be hexadecimal. The default is the current PE, if a file server is present. Otherwise, the default is the PE where the root file server is loaded.

2.3.2.73 `scp`

Display swap control structures.

```
scp [-l] address ...
```

Aliases:	NONE
`-l`	Include information about each `localCache`.
address	Swaps control structure address. Data at the specified address is formatted as if it were a `swap_control` structure.

A swap control structure describes an actor that is swapped.

2.3.2.74 `sem`

Display semaphore information.

```
sem [address ...]
```

Alias:	`locks`

address	Semaphore address. If omitted, all semaphores associated with waiting threads are displayed.

Semaphore addresses can be found in the `thSem` field, displayed by the `thread -l` directive.

2.3.2.75 sess

Display the session table entries.

```
sess [-l] [address ...]
```

Aliases:	NONE
`-l`	Generates a long report that includes `tty_sesslist` hash list information.
address	Session entry address. If omitted, all entries are displayed.

2.3.2.76 sio

Display `sio` (swap) server information.

```
sio [-b] [-l] [-m]

sio -m [-l] [-ll]
```

Aliases:	NONE
`-b`	Include bitmap assignments.
`-l`	Generate a long report. See directive forms for details.
`-ll`	Generate a longer report. See directive forms for details.
`-m`	Include information about the swap device.
`-s`	Include statistical information.

Directive forms are:

`sio [-b] [-l] [-s]`

Display `sio` server statistics.

`-b`	Include swap partition bitmap assignments.

`-l`	Include information about the swap partitions.
`-s`	Include swap partition statistics.

`sio -m [-l] [-ll]`

Display information about the swap device.

`-l`	Include a list of allocated and free areas.
`-ll`	Include a dump of the allocation map.

2.3.2.77 `sroute`

Display the special routes table (`MK_SROUTES_TABLE`) on the current PE.

```
sroute [-r] [-v] [pe_range ...]
```

Aliases:	`sroutes`, `srte`
`-r`	Include the raw image of the `MK_SROUTES_TABLE` entries.
`-v`	Include the address of the `MK_SROUTES_TABLE` entries.
pe_range	A range of PE numbers. If specified, routes from the current PE to and from these PEs are displayed. The default is all PEs. See Section 2.3.1.5, page 14, for details of the range format.

2.3.2.78 `stack`

Display kernel thread stack function call traces.

```
stack [-f frame_limit] [-l] [-n] [-r]
   [{ [-p] || [-P] || [-s stack_frame] || [pe_range:][slot_range] ... }]
```

Aliases:	`k`, `kernel`, `s`, `stk`
`-f` *frame_limit*	An integer that specifies the stack frame sanity check value, or the keyword `all`. When an integer value is specified, a message is displayed after that number of stack frames have been processed, asking if processing should continue. If the reply is yes, an additional *frame_limit* frames are processed. If the reply is no, processing is terminated. When `all` is specified, no sanity checking is done, and processing continues until the top of the stack is reached. The default value is 64.

`-l`	Include stack frame control information.
`-n`	Include the name of the nearest symbol or pool for stack arguments that appear to be addresses.
`-p`	Display panic stack using the panic stack frame pointer saved at `PanicFp` during panic processing.
`-P`	Display the stack pointed to by the `fp` register (`IR15`). The `fp` register is read from the logout area, and is available only if the PE has halted.
`-r`	Include the stacks of threads on remote sites that either created, or were created by the specified thread.
`-s` *stack_frame*	Address of a stack frame. If specified, the value is assumed to be a frame pointer. No validity check is made.
pe_range	A range of PE numbers. Numbers are assumed to be hexadecimal, and the trailing colon (`:`) is required. The default is the current PE.
slot_range	A range of thread entry (slot) numbers. The default is all threads.

See Section 2.3.1.5, page 14, for details of the range format.

The trace may not be accurate if a thread is connected to a PE.

Arguments may not be specified with the `-p`, `-P`, or `-s` options.

The `-p`, `-P`, and `-s` options are mutually exclusive.

Warning: Arguments passed to a function can be altered by that function. Thus, the values displayed by the `stack` directive may differ from their original values.

2.3.2.79 `status`

Display system status.

```
status [-H] [-n] [-P] -s [-t count] [-w]
```

Alias:	`stat`
`-H`	Display halt statuses as PEs are scanned. By default, a summary of halt statuses is displayed at the end of the scan.

`-n`	Omit scan for paniced or halted PEs. By default, all PEs are checked, and their halt / panic states are summarized.
`-P`	Display panic reasons as PEs are scanned. By default, a summary of paniced PEs is displayed at the end of the scan.
`-s`	Switch to the PANICed PE.
`-t` *count*	Include a list of the first *count* halted PEs, in ascending time order. The keyword `all` may be specified to display all halted PEs. The default value is 6.
`-w`	Include boot time of PEs that have been warmbooted.

PEs are scanned beginning with the highest numbered PE, in descending PE number order.

If neither the `-H` nor `-P` options are specified, an odometer showing the current count of halted / paniced PEs, and count of PEs scanned is displayed.

When the `-s` option is specified, the PANICed PE is chosen as follows:

- If the current PE has paniced and the PE was not paniced by another (i.e., the value of `panicPe` is equal to the current PE number), the current PE is selected.
- If the current PE was paniced by another, the PE initiating the panic is selected.
- Otherwise, the first PE found during the scan that was not paniced by another is selected.
- Otherwise, the first paniced PE found during the scan is selected.
- Otherwise, the first halted PE is selected.

PEs that have not halted are not included in the halt summary when the system image is a live system, since that is the normal state.

If the system image includes PEs that have halted, and either have not paniced, or have paniced but the content of `PanicFp` does not look like a stack frame pointer, then integer register 15 on that PE is evaluated. If its content looks like a stack frame pointer, a hint indicating that the content may be a stack frame pointer is displayed.

2.3.2.80 `swapper`

Display memory swap scheduler (swapper) information.

```
swapper [-e] [-i] [-l] [-o] [-Q] [-s] [slot-#]
```

Aliases:	NONE
`-e`	Display actors on all PEs.
`-i`	Display actor swap-in information.
`-l`	Include additional actor swap information.
`-o`	Display actor swap-out information.
`-Q`	Display actor swap in / out information.
`-s`	Display swapping statistics.
slot-#	Actor entry (slot) number.

By default, swapper statistics and actor swap in / out information (`-sQ` options) is displayed for all actors on the current PE.

2.3.2.81 `swapq`

Display processes currently swapped out by the `sio` (swap) server.

```
swapq [-l] [-ll] [-T] [proc_id ...]
```

Aliases:	NONE
`-l`	Include `tmCachePool` information about each swapped process.
`-ll`	Include addresses of `swap_control` structures.
`-T`	Display times in raw form (clock ticks). By default, times are convert to time of day.
proc_id	Process identifier. The form is [*pe_range*`:`][*slot_range*]

pe_range	A range of PE numbers. Numbers are assumed to be hexadecimal, and the trailing colon (`:`) is required. The default is the current PE.
slot_range	A range of process entry (slot) numbers. The default is all entries.

2.3.2.82 `syscalls`

Display system call statistics.

```
syscalls [-i]
```

Aliases:	NONE
`-i`	Calculates statistics for increment since last call.

2.3.2.83 `thread`

Display thread entries.

```
thread [-a] [-e] [-k] [-l] [-n] [-p] [-r] [-v]
   [ [pe_range:][slot_range] ...]

thread -i [-e] [-r] [ [pe_range:][slot_range] ...]
```

Alias:	`t`
`-a`	Include information about the actor associated with the specified thread.
`-e`	Display all threads on all sites. Ignored if arguments are specified.
`-i`	Display IPC traces. Shows threads that send or received an IPC. If the `-r` option is specified, the IPC is traced from the origination site to the terminating site.
`-k`	Include stack traces.
`-l`	Include thread details. If the `-a` and/or `-p` options are specified, actor and/or process details are included.
`-n`	With `-k`, includes the name of the nearest symbol or pool for stack arguments that appear to be addresses. With `-l`, includes the name of the nearest symbol or pool for link addresses.
`-p`	Include information about the process associated with the specified thread, if any.
`-r`	Include the stacks of threads on remote sites that either created, or were created by the specified thread.
`-v`	Include idle threads in the default display (that is, when no arguments are specified). Idle threads are those where `thStatus` is equal to `K_TIDLE` and `thIpc.itType` is equal to `K_THDLREPLIED`.

pe_range	A range of PE numbers. Numbers are assumed to be hexadecimal, and the trailing colon (:) is required. The default is the current PE.
slot_range	A range of thread entry (slot) numbers.

See Section 2.3.1.5, page 14, for details of the range format.

By default, a summary of threads on the current PE is displayed.

Only the `-e` and `-r` options have meaning when the `-i` option is specified. All others are ignored.

2.3.2.84 `timeout`

Display `callout` table entries.

```
timeout [-n]
```

Aliases:	`callout`, `calls`, `time`, `tout`
`-n`	Include pool name if the argument is a pool address.

2.3.2.85 `tpt`

Display tape device traces.

```
tpt [-L] [-t] [-T] [{ -a || device ... }]
```

Aliases:	NONE
`-a`	Display traces for all devices.
`-L`	Does not display locking information.
`-t`	Display timing information.
`-T`	Format table trace entries.
device	Name of the device.

If neither the `-a` option nor *device* is specified, a list of devices is displayed.

2.3.2.86 `traceback`

Display `kmem_alloc()` traceback.

```
traceback -v tb_wd ...
```

Alias:	`tb`
`-v`	Verbose. Include calling functions' argument list, when available. By default, only the function names are shown.
tb_wd	Traceback word. A traceback word contains addresses of the calling function pair: the function that called `kmem_alloc()` and its caller.

2.3.2.87 `traffic`

Display IPC traffic statistics.

```
traffic [-a] [-b] [-i] [-l] [-r] [-s] [-t] [server ...]
```

Aliases:	`ipc`
`-a`	Include call details for each server.
`-b`	Include bytes read and written.
`-i`	Calculate statistics for increment since last `traffic` display.
`-l`	Display a detailed list of IPCs sent by type.
`-r`	Display IPCs received.
`-s`	Display IPCs sent.
`-t`	Calculate statistics since boot time.
server	Server name. IPCs received by specified servers are displayed. This display is included when the `-a` option is specified.

If neither the `-r` nor `-s` options are specified, the default is `-rs`.

The `-i` and `-t` options are mutually exclusive. The default is `-t`.

The `-i` option is ignored if the system image is a dump or halted PE.

2.3.2.88 `ts`

Display the name of the symbol whose value is closest to but not greater than the address given.

```
ts [-a] address ...
```

Aliases:	NONE
`-a`	Shows all occurrences of *address*, if *address* is an unqualified symbol.
address	An address or symbol. See page 10.

If the symbol is within an actor, the symbol display includes the actor's name. The display is of the form *actor.symbol* if the actor is loaded, or *actor!symbol* if the actor is not loaded.

See also the `ds` directive, Section 2.3.2.19, page 26.

2.3.2.89 `tsconv`

Converts time stamp to date/time.

```
tsconv [-b] [-d] [-h] timestamp ...
```

Alias:	`rtc`
`-b`	Adjust low *timestamp* values by the boot time. This gives a reasonably accurate time-of-day value. Time stamps have low values from the time a PE is booted until the time of day is set in the kernel. These values represent time since boot.
`-d`	Include clock drift value for the current PE.
`-h`	Include clock rate for the current PE.

The time zone used depends on the *system-image*. For a live system, the local time zone where `crashmk(8)` is running is used. For a dump, the time zone where the dump was taken is used (based upon data in the dump header).

2.3.2.90 `tty`

Display `tty` table entries.

```
tty [-l] [ [-a] entry ...]
```

Aliases:	NONE
`-l`	Generates a long report that includes the `stty(1)` parameters.
`-a`	Specifies that *entry* arguments are addresses.

entry	A `tty` number, or if `-a` is specified, an address of a `tty` entry. If omitted, all opened entries are displayed.

2.3.2.91 `utrace`

Display kernel and server trace entries.

```
utrace [-b buf_hdr[,buf_hdr ...]] [-B buf_hdr[,buf_hdr ...]] [-h] [-i] [-I]
   [-K] [-L tracelist] [-n] [-p] [-P address] [-r range] [-S] [-T] [-U]
   [-v] [-x] [+] [count [server ...]]
```

Alias:	`ut`
`-b` *buf_hdr*	Define the address of a trace buffer. The form is *addr*[`.`[*server*][`.`*type*]]. Where *addr* is a buffer header address, *server* is the server name, and *type* is a trace type (`I`, `K`, `S`, or `U`). The default server name is taken from the name field within the header at *addr*. The default type is `U`. If a trace type flag is set in the trace header, it overrides both the specified or default value. This option is ignored when `+` is specified as an argument. By default trace buffer header addresses are found in the *namelist* Traces defined by the `-b` option are merged with those found in the *namelist*. This option may be useful when the *namelist* does match the system image, or the system image has been corrupted.
`-B` *buf_hdr*	Define address of a trace buffer header. The form is the same as that for the `-b` option. When the `-B` option is used only traces defined by the specified headers are displayed.
`-h`	Display trace header information.
`-i`	Reinitialize `utrace` definition tables. Definition tables are automatically reinitialized if the current PE has changed since the last time the `utrace` directive was invoked.
`-I`	Include `ITRACE` type entries.
`-K`	Include `KTRACE` type entries.
`-L` *tracelist*	Display traces defined by the list of `Trace` structures specified by *tracelist*. Forms are:

address	Address of the head of a list of `Trace` structures.
n	An ordinal (i.e. integer of 1 or greater). When this form is used, traces from the *n*th `badKmem` structure are displayed. A `badKmem` structure is created

when kmem corruption is detected, and contains the head of a list of `Trace` structures defining the traces that were saved. `badKmem` structures can be displayed via `kmem -d`.

`-n`	Include the name of the nearest symbol or pool for trace parameters that appear to be addresses.
`-p`	Include PAL trace entries, merged in approximate time order.
`-P` *address*	Address of initial MKPAL trace entry. If specified, the `-p` option is implied. If not specified, `crashmk` attempts to determine the address. This option is ignored when `+` is specified.
`-r` *range*	Sets the search range to *range*, an integer. The initial default value is 20. After *range* has been specified, it becomes the default for subsequent `utrace` directives until overridden. The search process (see Section 2.3.2.91.1, page 78) is disabled when *range* is set to zero, or when the system image is of a live system.
`-S`	Include `STRACE` type entries.
`-T`	Display time stamp in raw form. By default, the time stamp is converted to time of day.
`-U`	Include `UTRACE` type entries.
`-v`	Include the entry's address.
`-x`	Excludes specified trace types.
`+`	Continue trace from next oldest entry. This specification is ignored when the *system_image* is a live system.
count	Number of entries to display (default: 24).

The following keywords are also allowed:

`all`	Display all entries.
`scan`	Scans all entries, and notes ones that are out of sequence (static system image only).

server	Display entries generated by specified servers, only.

Entries are displayed in descending chronological order.

If no trace types are specified, all trace types (`ITRACE`, `KTRACE`, `STRACE`, and `UTRACE`) are displayed.

Trace definitions made by the `-b`, `-B`, or `-L` options remain in force until either the current PE has changed, or the `-i` option is specified.

Trace types are:

`ITRACE`	Tend to be from clock driven interrupt activities.
`KTRACE`	Kmem-related activities.
`STRACE`	Swap-related activities.
`UTRACE`	All other activities.

Caution: When the *system_image* is of a live system, trace entries are updated continuously. Thus, while a best effort is made, the merge of trace entries for the kernel and servers may not be accurate. This is particularly true for the merge of MKPAL trace entries.

2.3.2.91.1 Search Range

Trace entries for each server and trace type within servers are written into separate circular buffers. The offset of the most recent entry written, *first_index*, is maintained in the trace buffer's header. If data has not been flushed from cache, one of the following conditions may exist:

- *first_index* is in cache and the latest entry is in memory, then the value is too low.
- *first_index* is in memory and the latest entry is in cache, then the value is too high.
- *first_index* and the latest entry are both in cache, then the value may be either too high or too low.

`utrace` attempts to overcome this by searching for the entry with the highest RTC value in a range beginning at *first_index* - *range* and extending through *first_index* + (*range* * 2). This entry is assumed to be the most current trace entry. The lowest RTC value in the current cycle through the trace buffer, *low_RTC*, is assumed to be the RTC value in the next entry beyond the one with the highest RTC.

When entries are displayed, two passes are made through the trace buffer. On the first pass, only those entries that have been written during the current cycle are displayed. That is, entries with an RTC value that is greater than or equal to *low_RTC*. Entries with a lower RTC value are displayed on the second pass.

When entries are skipped on either pass, an informational message is displayed noting the number of entries that were skipped.

This process is intended to eliminate or reduce the instances of out of sequence trace entries that can appear when a PE's cache has not been flushed.

2.3.2.92 `var`

Display system variables (`var` structure).

```
var [actor ...]
```

Aliases:	NONE
actor	Specifies the actor whose `var` structure is to be displayed.

If *actor* is not specified and a default actor has been set by the `default actor` directive, the `var` structure for that actor is displayed. If no default actor has been set, the `var` structure for the file server is displayed.

If *actor*, or default actor, is loaded on the current PE, its `var` structure is displayed. Otherwise, the `var` structure on the first PE where the actor is loaded is displayed.

Note: Many servers have a `var` structure, which is initialized by the configuration server. Thus, all `var` structures get the same initial values (for example, any server's `var` structure indicates the correct number of entries in the file table). The only difference in the per server `var` structures will be that some servers go on to dynamically allocate tables and initialize the `vb_`*xxx* and `ve_`*xxx* entries (for example, the `vb_file` and `ve_file` entries in the file server's `var` structure will be non-zero).

2.3.2.93 `vfs`

Display the mount table in `vfs` chain order.

```
vfs [-e] [-w] [pe_range ...]
```

Aliases:	NONE
`-e`	Display all in use entries on all PEs where a mount table exists.
`-w`	Include a list of PEs where a file server is loaded.

pe_range	A range of PE numbers. Numbers are assumed to be hexadecimal, and the trailing colon (`:`) is required. The default is the current PE if it has a file server. Otherwise, the default is the root file server's PE. Numbers of PEs that do not have file servers are silently ignored. See Section 2.3.1.5, page 14, for details of the range format.

VFS entries are contained within mount table entries, and are found only on PEs with a file server.

If the `-w` option is specified without arguments, only file server information is displayed. Otherwise, if no arguments are specified, the default is to display all in use entries on the default PE.

Arguments are ignored when the `-e` option is specified.

2.3.2.94 `vmlock`

Display vmlock traces.

```
vmlock [-i] address ...

vmlock -a [-i]
```

Aliases:	NONE
`-a`	Display traces for all `localCache` structures on the current PE.
`-i`	Include `vmlock` debug information.
address	Address of a `localCache` structure. Data at this address will be formatted as if it were a `localCache` structure. No validation is done.

Arguments are ignored when the `-a` option specified.

2.3.2.95 `vnode`

Display vnodes.

```
vnode [-a] [-e] [-l] [-ll] [-w] [ [pe_range:][entry] ...]
```

Alias:	`vn`

`-a`	Include unused entries. This option is assumed when arguments are specified.
`-e`	Display all in use entries on all PEs where a `vnode` table exists.
`-l`	Generates a long report.
`-ll`	Include `vnode` trace buffer.
`-w`	Include a list of PEs where a file server is loaded.
pe_range	A range of PE numbers. Numbers are assumed to be hexadecimal, and the trailing colon (`:`) is required. The default is the current PE if it has a file server. Otherwise, the default is the root file server's PE. Numbers of PEs that do not have file servers are silently ignored.
entry	`vnode` entry. Can be either of the following:

slot_range	A range of `vnode` entry (slot) numbers.
address	Address of a `vnode` entry. The address is ignored if it is not within the `vnode` table. Otherwise, the entry nearest to the address from below is displayed.

See Section 2.3.1.5, page 14, for details of the range format.

`vnodes` are contained within `inode` structures, and are found only on PEs with a file server.

`vnodes` that are within `prnodes` (`/proc` inodes) can only be displayed via the `prnode` directive.

If the `-w` option is specified without arguments, only file server information is displayed. Otherwise, if no arguments are specified, the default is to display all in use entries on the default PE.

Arguments are ignored when the `-e` option is specified.

If no arguments are specified, a one-line summary of all active `vnode` entries is displayed.

2.3.2.96 `walk`

Walk a linked list and verify list integrity.

```
walk [-c [d_off,]count] [-d] [-m mode] [-n] [-o l_off] [-p] [-v] start
   [#entries]
```

Aliases:	NONE
`-c` [*d_off*`,`]*count*	Include a dump (similar to that of `od`) of the data from each list entry.
	d_off — Entry relative byte offset of the start of the data values to be dumped. The trailing comma (`,`) is required. The default is zero. The prefix `w:` can be used to indicate a word value, which is converted to bytes.
	count — Number of data values to be dumped. The size of the data item is determined by the display mode.
`-d`	Link is a `dlink` structure. Verify both `dnext` (forward link) and `dprev` (backward link).
`-m` *mode*	Display mode for dumped data. The default is the current display mode. See Section 2.3.1.3, page 12, for valid *mode* values.
`-n`	Include the name of the nearest symbol of pool for dumped data that appear to be addresses. This option is ignored if the `-d` option is not specified.
`-o` *l_off*	Entry relative byte offset of the link field, or in the case of a `dlink` structure, the entry relative byte offset of the structure. The field is expected to be word aligned. The default is zero. The prefix `w:` can be used to indicate a word value, which is converted to bytes.
`-p`	Walk `dlink` structure `dprev` list. This option implies the `-d` option.
`-v`	Verbose. Include a one-line summary of each entry.
start	Starting address. Can be either of the following forms:
	address — Address value.
	`+` — Continue from the last address displayed.
#entries	Maximum number of entries. This is an upper bound for the number of entries to process. The default is all entries.

The walk is terminated upon any of the following conditions:

- The maximum number of entries have been processed.

- The list folds back upon itself.
- A next pointer address is bogus.
- An end-of-list condition is found.

  Normal end-of-list conditions are:

  – The next pointer is null.

  – The next pointer address is the start address.

2.3.2.97 xddtab

Display `xdd_tab` structures.

```
xddtab [-a] [-b] [-B buflim] [-e] [-l] [-s] [-w] [ [pe_range:][entry] ...]
```

Alias:	`xdd`
`-a`	Include unused entries. This option is assumed when arguments are specified.
`-b`	Include queued and active I/O buffer chains.
`-B` *buflim*	An integer that specifies the buffer chain sanity check value, or the keyword `all`. When an integer value is specified, a message is displayed after that number of buffer chain entries have been displayed, asking if processing should continue. If the reply is yes, an additional *buflim* entries are displayed. If the reply is no, processing of the current chain is terminated. When `all` is specified, no sanity checking is done, and processing continues until a NULL link or bogus address is found. The default is 128 entries. When specified, the `-b` option is implied. The specified value becomes the default until another is specified, or the PE changes.
`-e`	Display all in use entries on all PEs where a `xdd` table exists.
`-l`	Include a long version of the `xdd_tab` structures. By default, only a summary is displayed. The long format includes all fields of the structure, but only the addresses of the `xdd_stat` sub-structure.
`-s`	Include the `xdd_stat` sub-structure.
`-w`	Include a lit of PEs where a disk server is loaded.

pe_range	A range of PE numbers. Numbers are assumed to be hexadecimal, and the trailing colon (`:`) is required. The default is the current PE if a disk server is loaded. Otherwise, only a list of PEs where disk servers are loaded is displayed.
entry	Xdd table entry. Can be either of the following:
	slot_range — A range of `xdd_tab` entrie (slot) numbers.
	address — `xdd_tab` entry address. The address is ignored if it is not within the `xdd` table. Otherwise, the entry nearest the address from below is displayed.

Xdd tables are found only on PEs with a disk server.

If the `-w` option is specified without arguments, only disk server information is displayed. Otherwise, if no arguments are specified, the default is to display all active entries on the current PE.

Arguments are ignored when the `-e` option is specified.

2.3.3 `crashmk` Directives and Aliases

The following table shows the `crashmk` directives and their aliases, if any.

Table 5. Directives and Aliases

Directive	Aliases
`!`	
`.`	
`actor`	`a`
`address`	`addr, addrof`
`bits`	
`buffer`	`b`
`bufhdr`	`buf, hdr`
`cdcmp`	`ccmp, code, codecmp`
`chtab`	`ch, cht, chtable`
`cku_private`	`cku_priv, ckup`

Directive	Aliases
cm_conn	cmc
cm_serv	cms
cmon	
cnode	cn
config	conf
console	cons
ctx	
debug	
default	def, set
dfsstat	
dmptab	
dnlc	
ds	
errec	erec, erlog
eslice	esl
file	f, files
find	scan, search
flow	
gtb	
help	?, h, man
info	
job	
kftrace	kft
kmem	
kstack	kstk, kvs
lkup	grep, nm, syms
loadtable	lt
logout	log
map	

Directive	Aliases
mdw	mdws, msd, msds
mem	
mmu	
mount	m, mnt
msgq	mq
nc1inode	nc
nfsmi	
od	
onpe	onpes
paltrace	pt
pddtab	pdd
pemap	pem
pequeue	peq
pmfsst	pmfs, pmstats
poke	
pool	pools
pput	pages
prnode	prn
proc	p, ps
psched	
pty	
pwho	pwhos
quit	exit q
redirect	>, >>, redir
reg	regs
reslist	
rlimits	rlimit, rlim
rmon	
rnode	rn

Directive	Aliases
rnodetab	rnt
runq	rq
rwlock	
sar	
scp	
sem	locks
sess	
stack	k, kernel, s, stk
status	stat
swapper	
swapq	
syscalls	
thread	t
timeout	call, callout, calls, time
	tout
tpt	
traceback	tb
traffic	ipc
ts	
tsconv	rtc
tty	
utrace	ut
var	
vfs	
vmlock	
vnode	vn
walk	
xddtab	xdd

crashmk Structures [3]

This chapter includes the following:

- Diagrams of UNICOS/mk data structures displayed or used by selected directives
- Notes on the interpretation of selected directives' output

3.1 actor

Displays actor information.

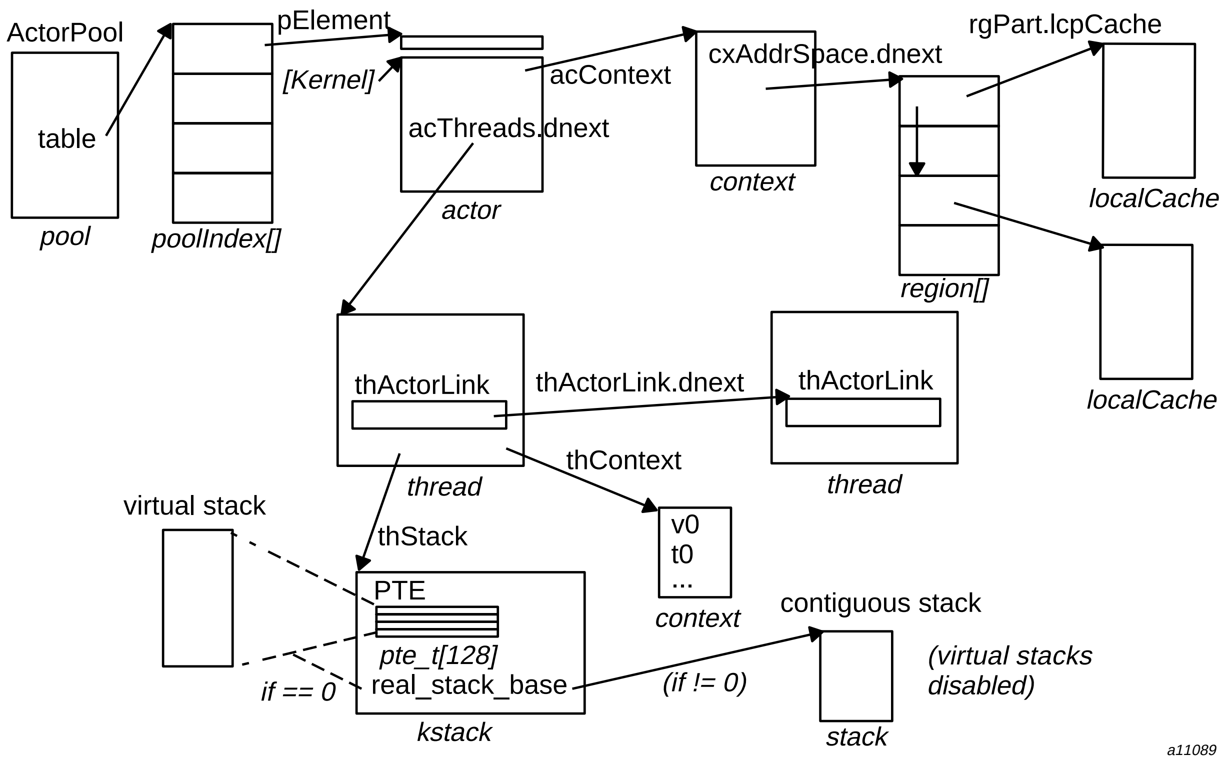

Figure 1. Actor and Associated Structures

3.2 bufhdr

Displays system buffer headers.

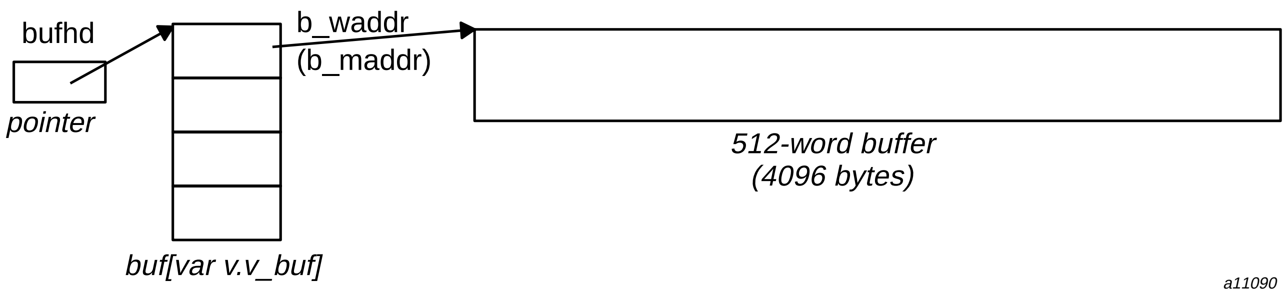

Figure 2. `bufhdr` and Buffer Structures

3.3 chtab

Displays NFS client handle entries.

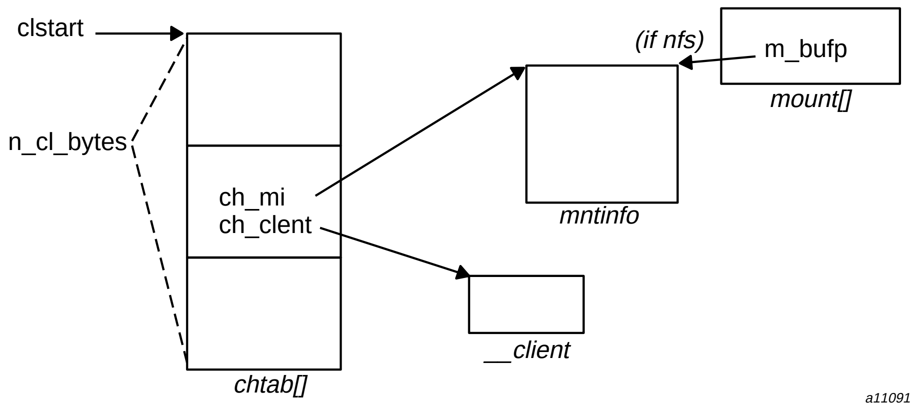

Figure 3. `chtab` Structures

3.4 cku_private

Displays NFS `cku_private` structures.

`cku_private:  Private data per rpc handle.` This directive simply casts any given address as a `cku_private`.

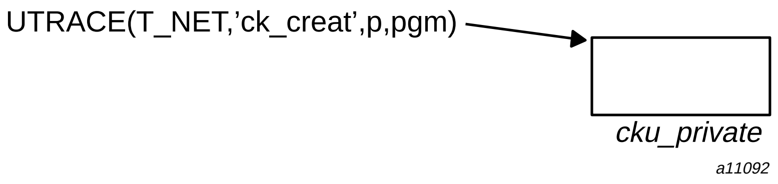

Figure 4. `cku_private` Structures

3.5 config

Displays `config` server information.

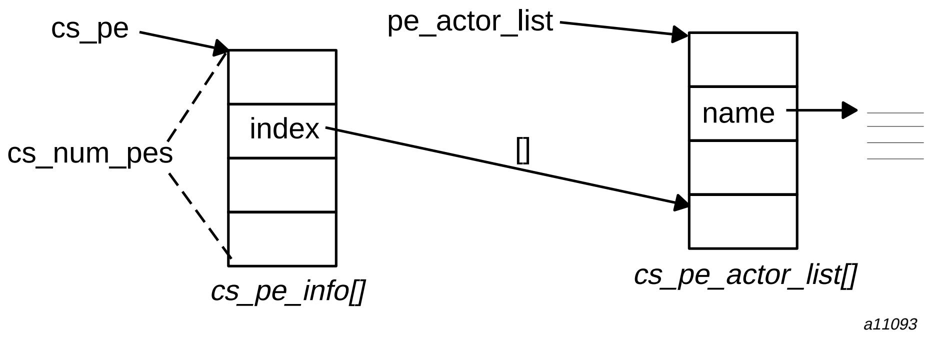

Figure 5. `config` Structures

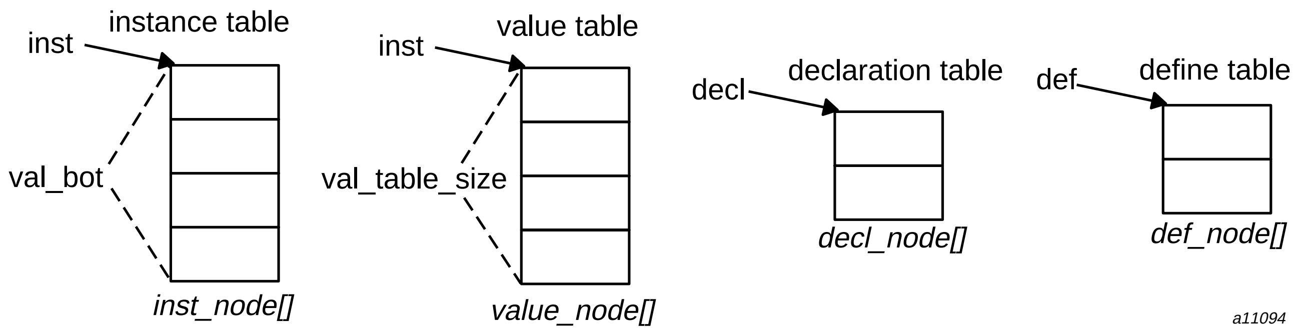

Figure 6. `config -s` Structures

3.6 `console`

Displays console buffers.

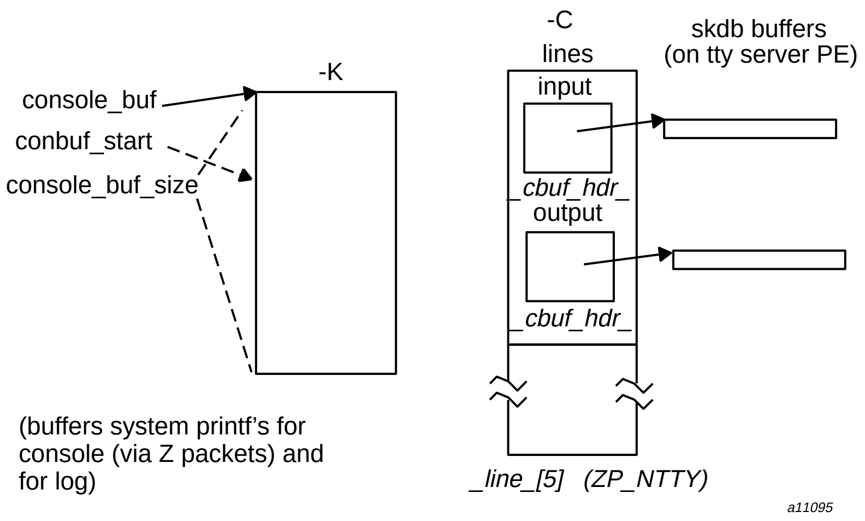

Figure 7. `console` Structures

3.7 `ctx`

Displays thread context (`KnThreadCtx`) structure values.

KnTrheadCtx is a typedef of the regsalpha structure. Also see the stack and thread directives' diagrams of the thread and associated structures. The following structure exists at the base of the thread structure: structure SupThreadDesc

```
0: __b_dlink struct dlink
0x10: sysCtx: pointer to struct regsalpha
0x18: userCtx: pointer to struct regsalpha
0x20: intrCtx: pointer to struct regsalpha
0x28: fpuCtx: pointer to array of [32] of unsigned int
0x30: ERegPtr: pointer to struct ERegisters
```

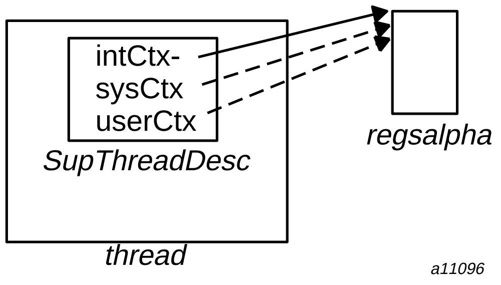

Figure 8. ctx Structures

3.8 dmptab

Displays PE dump table pointer (struct DmpTabPtr) and its associated dump table entries (typedef DmpTab).

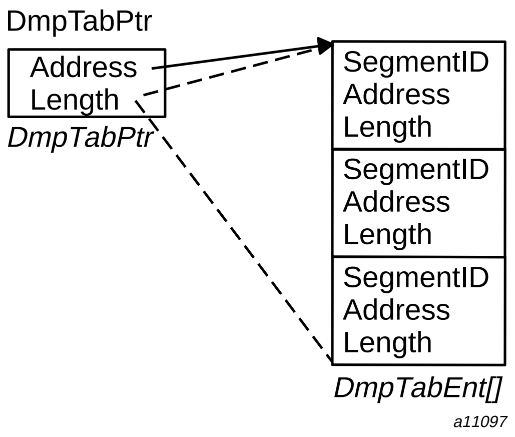

Figure 9. `dmptab` Structures

3.9 ds

Displays the name of the symbol whose value is closest to, but not greater than, the specified address, and the offset from that symbol.

Example 1: ds address

Displays a symbol and offset.

```
0x001> ds 0x021c7f
       ipcMsg.o + 0x54d
```

The `ds` directive knows about certain internal structures. If the address is within one of these structures, the symbol and offset is followed an equal sign, the structure name, and the offset into that structure. For example:

```
0x001> ds 0xfffffc0007d45728
       bufhd + 0x6c45728 = &amp;Proc[1]+0x510

0x000> ds 3ffffff
        GPM.edata + 0x311be07
          536 (0x218) byte kmem segment: 0x3fffd80 + 0x27f
            allocated by:pool::reallocateIndex(int) , pool::allocate(int)
```

Example 2: ds `-m` address

Displays a non-demangled symbol and offset.

```
0x001> ds -m 0x021c7f
       ipcMsg$cipcMsg.o + 0x54d
```

Example 3: ds -a address

Shows all occurences of *address*, if *address* is an unqualified symbol.

```
0x001> ds -a ut
        kernel.ut + 0
        PM.ut + 0
        fsa!ut + 0
        log!ut + 0
        info!ut + 0
        config!ut + 0
        packet.ut + 0
        disk.ut + 0
        tty!ut + 0
        netdev.ut + 0
        file.ut + 0
        socket.ut + 0
        alm!ut + 0
        grm!ut + 0
        sio.ut + 0
        GPM!ut + 0
```

Note: the *server***.***symbol* versus *server***!***symbol* notation. "." is used if the *server* is loaded on the current PE. "!" is used if it is not.

3.10 eslice

Displays `eslice` structures.

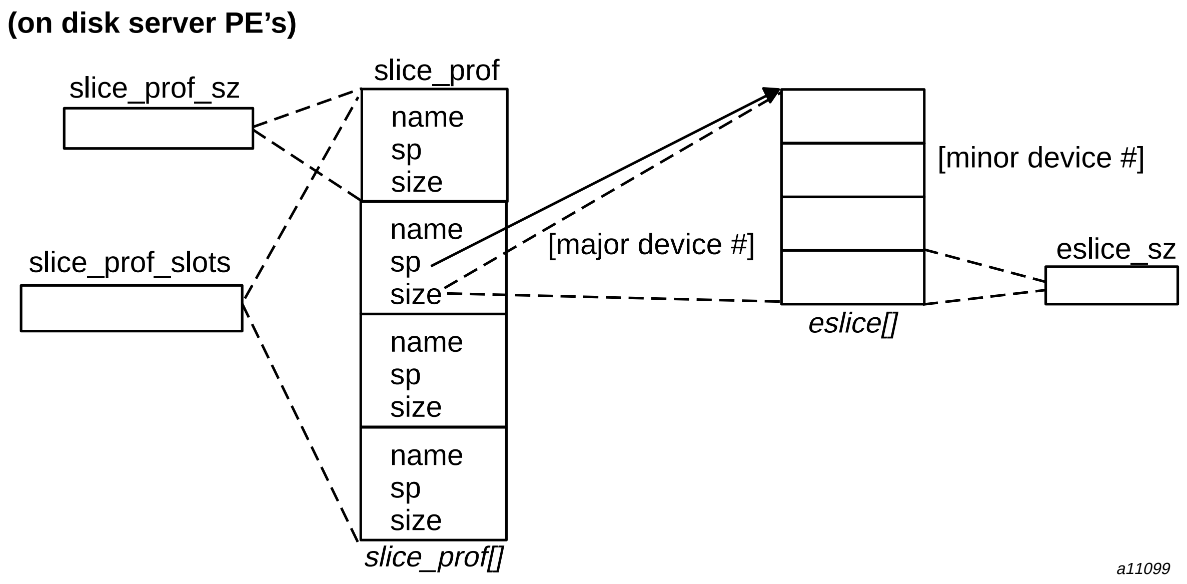

Figure 10. `eslice` Structures

3.11 `file`

Displays file table entries.

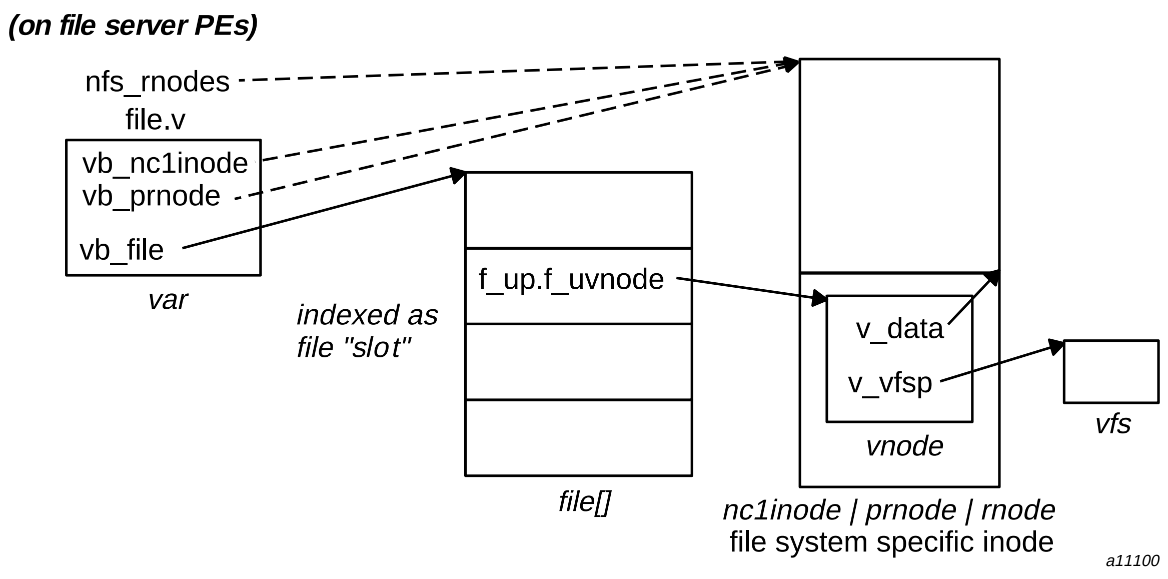

Figure 11. File Structures

3.12 `flow`

Controls FlowTrace operation. Also see the `kftrace` directive.

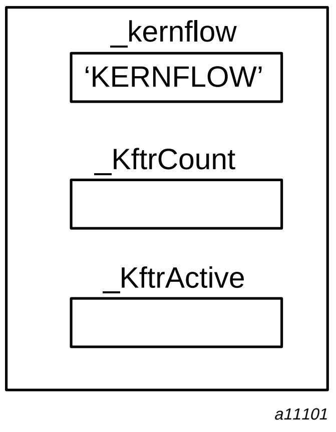

Figure 12. `flow` Words

3.13 info

Displays information server cache.

Example 4:

30/srv/info/export/com/INFOsCache.h:

```
struct InfoId {
        int             *name;          /* info type Id */
        int             groupId;        /* info group */
        SrvId           *si;            /* ID of server it came from */
        int             resync:32;      /* resync age for cache (secs) */
        int             timestamp:32;   /* age of data in buffer */
        int             namelen:4;      /* # of name levels */
        int             flags:60;       /* from INIT request. See infos.h */
        char            *buf;           /* pointer to data */
        int             recsize:32;     /* Size of info record */
        int             numrec:32;      /* Number of records */
        int             requests:32;    /* count of requests to server */
        int             failures:32;    /* failed request to server */
        struct InfoId   *next_server;   /* Next server's info */
        struct InfoId   *list;          /* Next info in request chain */
        struct InfoId   *next_subnode;  /* Next subnode off this node */
};
```

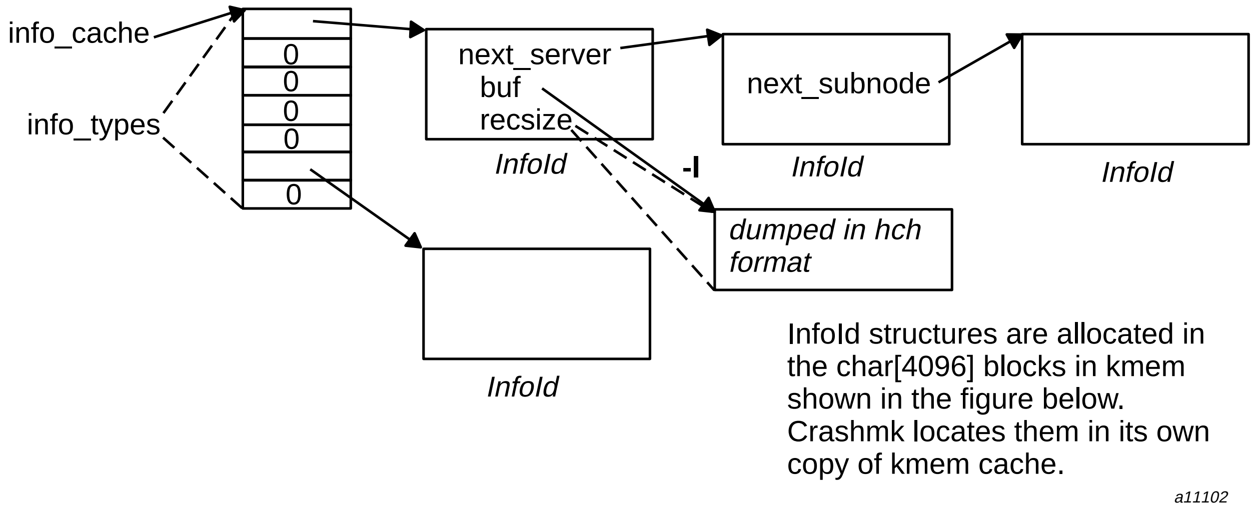
info_cache
info_types
0
0
0
0
0
next_server
buf
recsize
InfoId
-l
next_subnode
InfoId
InfoId
dumped in hch format
InfoId
InfoId structures are allocated in the char[4096] blocks in kmem shown in the figure below. Crashmk locates them in its own copy of kmem cache.
a11102

Figure 13. InfoId Structures

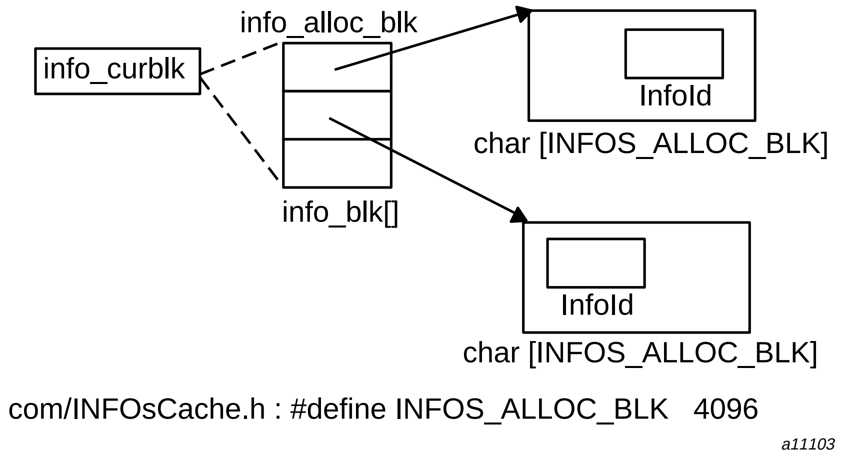

Figure 14. InfoId Structures Allocated in `kmem`

3.14 `job`

Displays Global Job Table information.

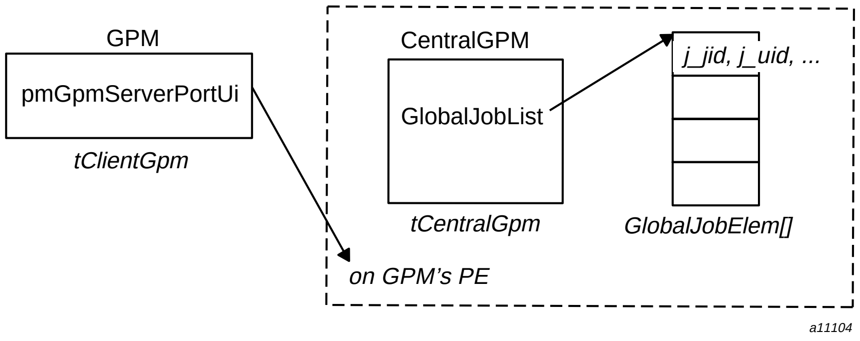

Figure 15. GPM Structures

3.15 `kftrace`

Displays the kernel FlowTrace information.

Example 5: `struct`

Structure of a trace entry.

```
struct tnbstruct {
        int:    32;                     /* word 0 */
        int:    16;
        int     namelen:16;
        struct tnbstruct *thisptr;      /* word 1 */
        unsigned long numcalls;         /* word 2 */
        unsigned long totaltime;        /* word 3 */
        struct tnbstruct *nextptr;      /* word 5 */
        char    name[MAXNAMEWORDS*8];   /* words  6, 7, 8 */
        int     avgtime;                /* word 9 */
        struct kft_info *kftlink;       /* word 10 (Extended flowtrace only) */
};
```

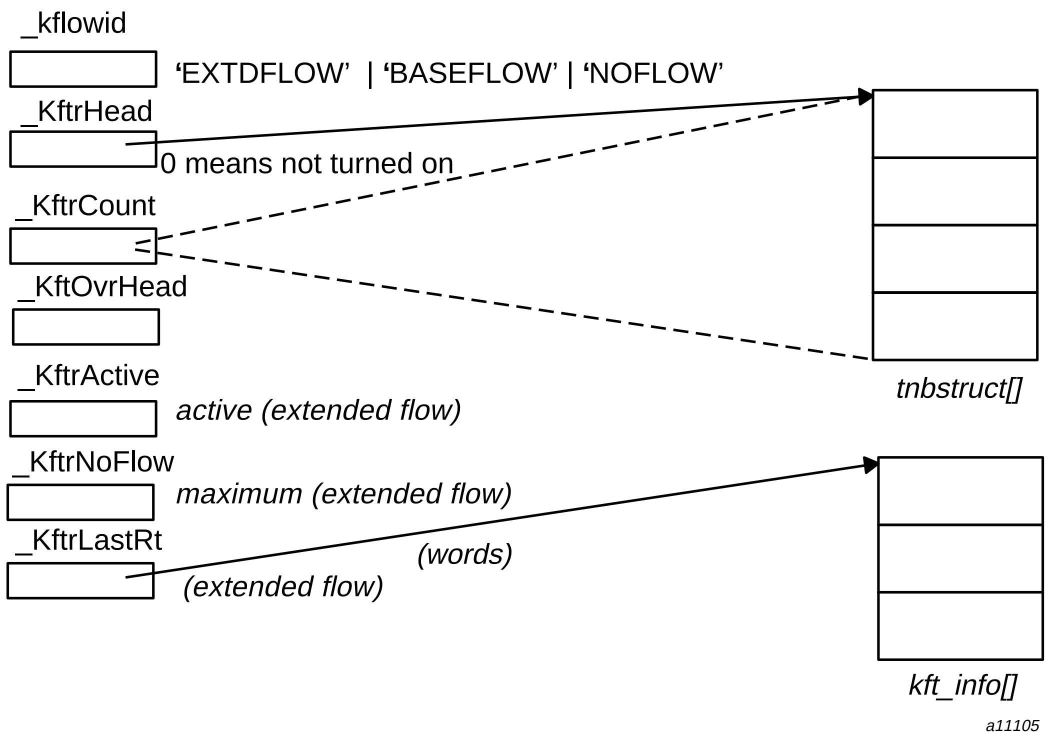

Figure 16. `kftrace` Structures

With extended flow, _KftrLastRt points to kft_info structures, which are defined as:

```
struct kft_info {
        uint    kft_tnb0;             /* tnb for caller */
        uint    kft_tnb1;             /* tnb for callee */
        uint    kft_ncalls;           /* number of calls */
        uint    kft_time;             /* Real time clocks */
        struct  kft_info *kft_link;   /* link to next kft_info for tnb */
};
```

3.16 kmem

Displays kernel memory information.

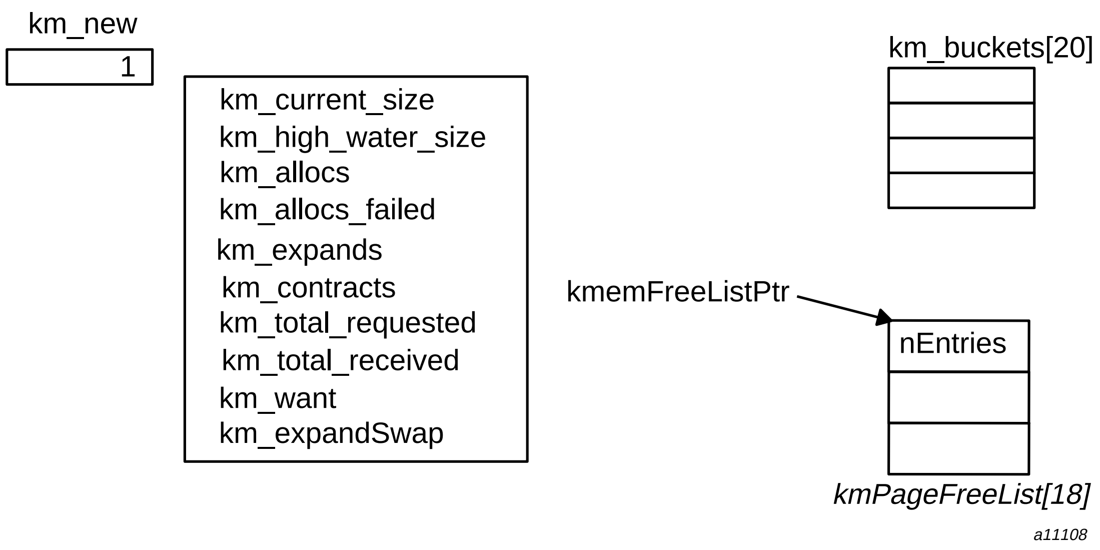

Figure 17. kmem Basic Information

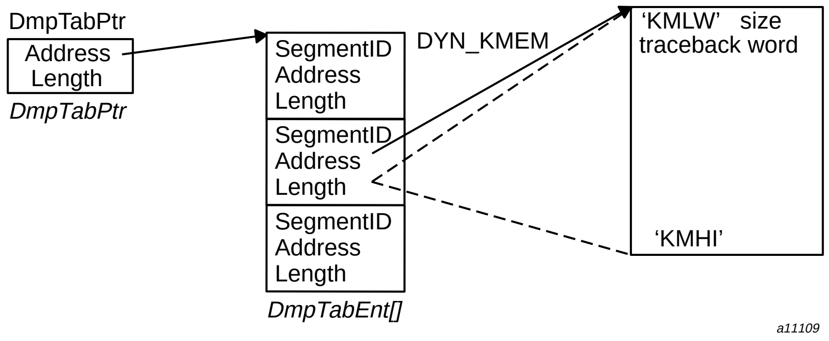

Figure 18. `kmem` Call Chain Information

3.17 `kstack`

Displays virtual kernel stack information. Also see the `> stack` directive.

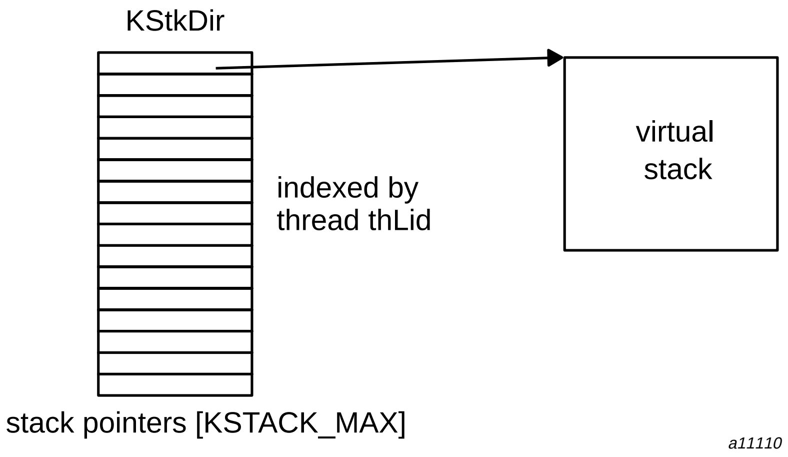

Figure 19. `kstack` Structures

3.18 `loadtable`

Displays boot actor load table.

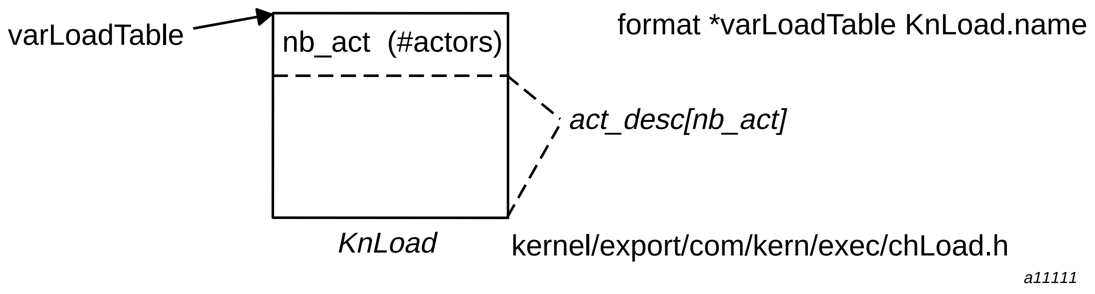
varLoadTable
nb_act (#actors)
format *varLoadTable KnLoad.name
act_desc[nb_act]
KnLoad
kernel/export/com/kern/exec/chLoad.h
a11111

Figure 20. Loaded Actors Table

3.19 logout

Displays the logout structure. Internal registers are copied to the logout structure when a PE halts.

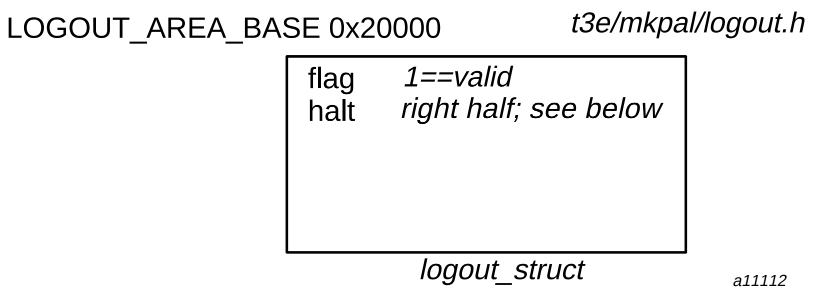

Figure 21. Logout Reasons

The logout flag, in `logout_area.flag`, is zero when a PE running normally. When a PE halts, its value goes from 1 (start of halt process) to 0xff (PE halted), in defined steps. The following table shows the values of the logout flag, and which sections of the logout area have been saved to that point.

```
Flag value | Sections saved
-----------+---------------------------------------------------------
    1      | * halt code
           | * logout version
-----------+---------------------------------------------------------
    2      | * PAL shadow registers
```

```
           | * scalar registers
           | * floating point registers
-----------+-----------------------------------------------------------
    3      | * EV5 IPRs
           | * EV5 CBOX registers
           | * EV5 DTB/ITB
-----------+-----------------------------------------------------------
    4      | * C-option registers
           | * 1st portion of MMRs
-----------+-----------------------------------------------------------
    5      | * M-option registers
           | * 2nd portion of MMRs
-----------+-----------------------------------------------------------
   0x10    | * 3rd portion of MMRs
-----------+-----------------------------------------------------------
   0x20    | * Kernel E-registers
-----------+-----------------------------------------------------------
   0x40    | * User E-registers
-----------+-----------------------------------------------------------
   0x50    | notify boot PE that this PE is halting
-----------+-----------------------------------------------------------
   0x60    |
-----------+-----------------------------------------------------------
   0x70    | * 1st portion of R-option registers
           |   . r_lwho, r_mon[0-3]
-----------+-----------------------------------------------------------
   0x78    | * BESU bar_cfg registers
-----------+-----------------------------------------------------------
   0x80    | * BESU bar_ctl registers
-----------+-----------------------------------------------------------
   0x90    | flush cache, spin-wait for boot PE PROD of other PEs
-----------+-----------------------------------------------------------
   0xa0    | * remainder of R-option registers
-----------+-----------------------------------------------------------
   0xff    | * I-option registers
           |   Note: I-option registers are saved only when r_chan7[0:0]
           |         is zero
-----------+-----------------------------------------------------------
           | flush cache, halt
-----------+-----------------------------------------------------------
```

The machine check area is filled whenever a machine check occurs. Machine checks may not result in a system halt, although in most cases they do. By default the machine check area will be displayed only when there is a logout halt code, and it is a machine check halt.

The following structure, within `crashmk`, is used to map the halt code saved in `logout_area.halt` to the correstonding halt reason string. The fields are `#define`d halt code value, halt reason string, and machine check indicator. A non-zero value indicates the halt is due to a machine check.

```
HaltDesc HaltReason[] = {
    {0,                     "Not halted", 0},
    {MKPAL_HALT_SW_HALT,    "Kernel executed a halt instruction", 0},
    {MKPAL_HALT_MEM_MGMT,   "Memory management fault while in MKPAL", 0},
    {MKPAL_HALT_UNALIGNED,  "Unaligned fault while in MKPAL", 0},
    {MKPAL_HALT_OPCDEC,     "Reserved instruction while in MKPAL", 0},
    {MKPAL_HALT_ARITH,      "Arithmetic fault while in MKPAL", 0},
    {MKPAL_HALT_MCHK,       "Machine check", 1},
    {MKPAL_HALT_MCHK_PAL,   "Machine check while in MKPAL", 1},
    {MKPAL_HALT_MCHK_DBL,   "Double machine check", 1},
    {MKPAL_HALT_FEN,        "FEN while in MKPAL", 0},
    {MKPAL_HALT_BPT,        "Breakpoint while in MKPAL", 0},
    {MKPAL_HALT_SYSCALL,    "Syscall from within the kernel", 0},
    {MKPAL_HALT_DFAULT,     "Dfault while in MKPAL", 0},
    {MKPAL_HALT_UNEXPECT,   "Unexpected interrupt", 0},
    {MKPAL_HALT_BUGCHK_PAL, "Bug check in MKPAL", 0},
};
```

The halt code values are `#define`d in `mkpal/interface.h`, as follows:

```
MKPAL_HALT_SW_HALT      1       /* OS executed a halt instruction */
MKPAL_HALT_MEM_MGMT     2       /* Memory management fault while */
MKPAL_HALT_UNALIGNED    3       /* Unaligned fault while executing */
MKPAL_HALT_OPCDEC       4       /* OPCDEC while executing in MKPAL */
MKPAL_HALT_ARITH        5       /* Arithmetic fault while executing */
MKPAL_HALT_MCHK         6       /* Machine check */
MKPAL_HALT_MCHK_PAL     7       /* Machine check while in MKPAL */
MKPAL_HALT_MCHK_DBL     8       /* Double machine check */
MKPAL_HALT_FEN          9       /* FEN while executing in MKPAL */
MKPAL_HALT_BPT          10      /* Breakpoint while in MKPAL */
MKPAL_HALT_SYSCALL      11      /* System call from Kernel mode */
MKPAL_HALT_DFAULT       12      /* DFAULT while in MKPAL */
MKPAL_HALT_UNEXPECT     13      /* Unexpected interrupt was received */
MKPAL_HALT_BUGCHK_PAL   14      /* Bug check in MKPAL */
```

3.20 map

Displays system map structures.

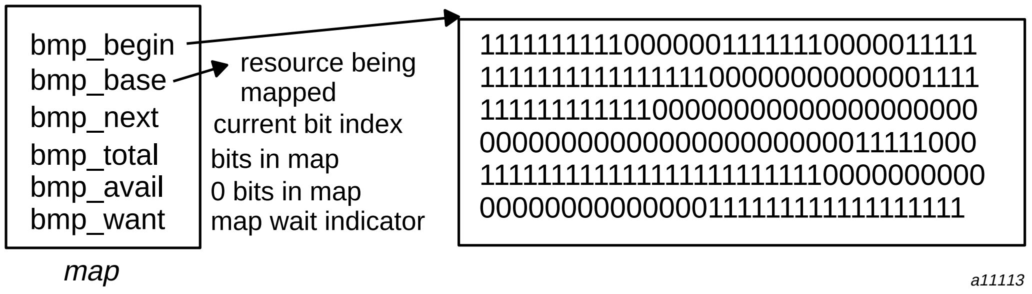

Figure 22. map Structures

3.21 mem

Displays layout of actors in memory. (Not supported on Cray T3E system.)

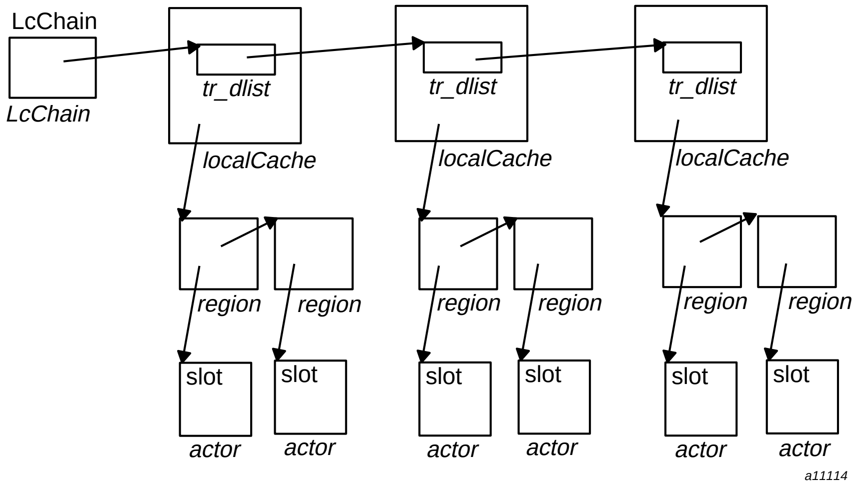

Figure 23. Local Cache Chain (> Mem).

3.22 mmu

Displays the Memory Management Unit (MMU) translation data (Cray T3E system only).

You must already know the address of a mmu_t structure to use this directive.

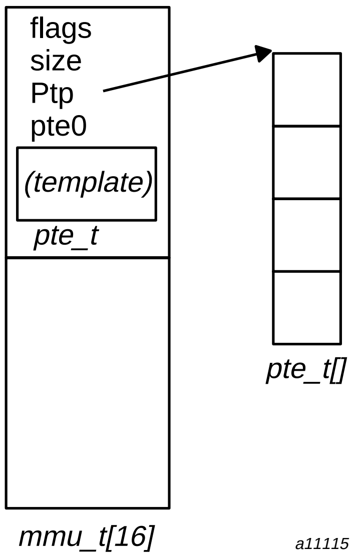

Figure 24. mmu Structures

3.23 mount

Displays mount table entries.

When there are multiple file servers, one file server is the root server. The mount table on the root server's PE contains all mounted devices. The mount table on non-root server PE's contain the root device and all devices mounted in the path to the local devices.

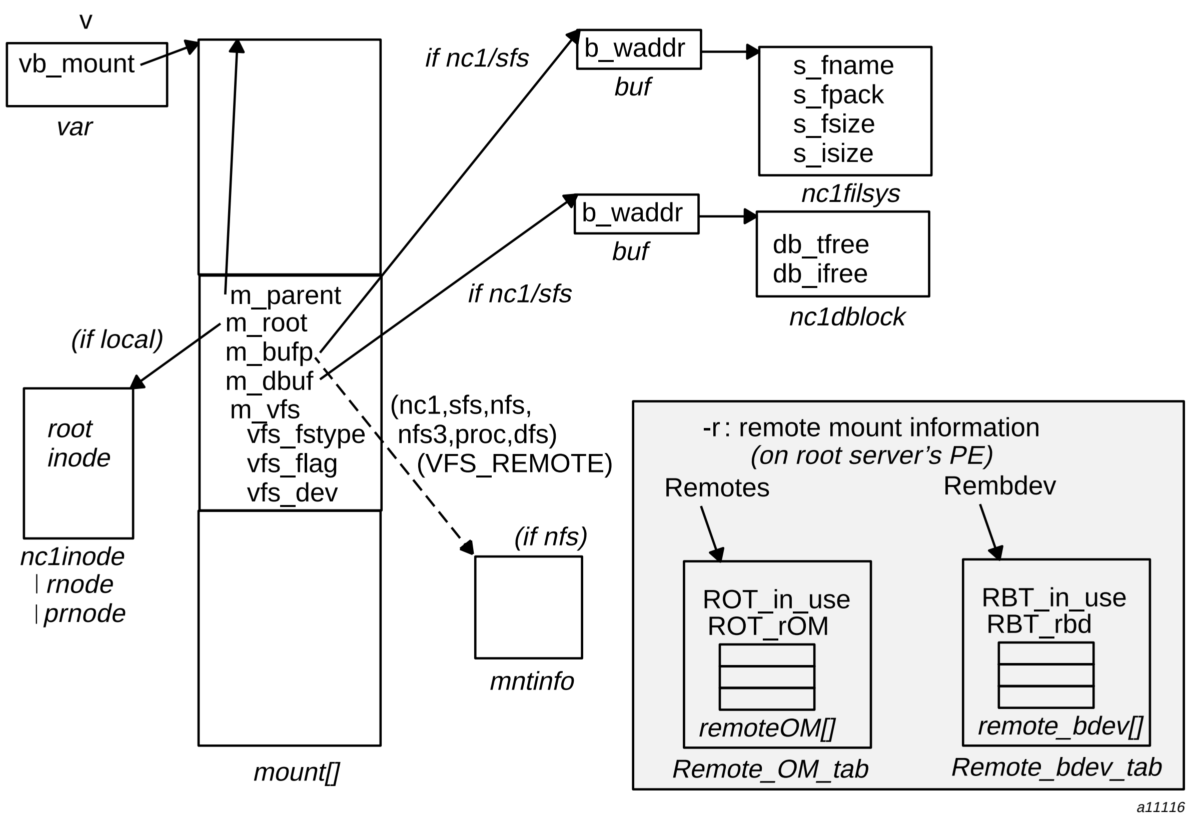

Figure 25. `mount` Structures

3.24 `nc1inode`

Displays `nc1inode` table entries.

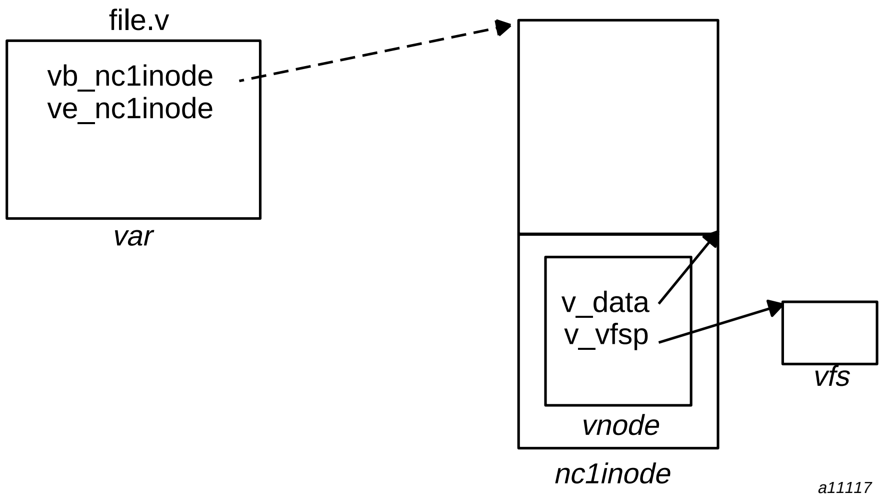

Figure 26. nc1inode Structures

3.25 nfsmi

Displays NFS mount information (mntinfo) structures.

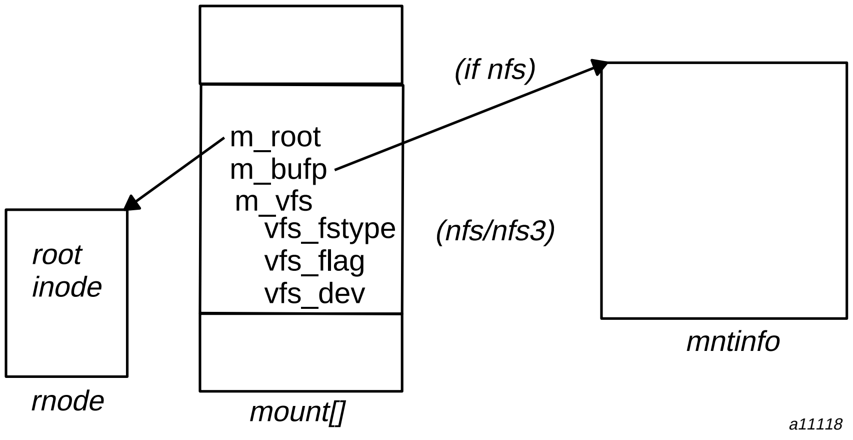

Figure 27. nfs Mount Info Structures

3.26 paltrace

Displays MKPAL trace entries.

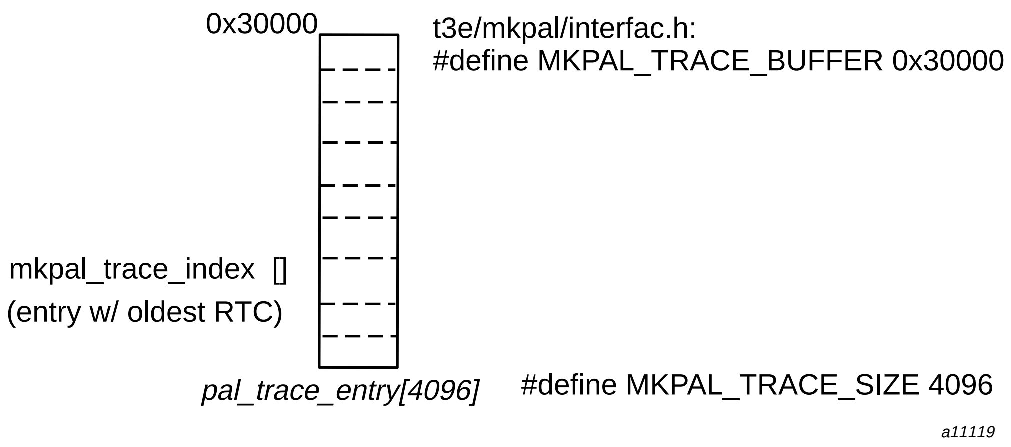

Figure 28. paltrace Structures

3.27 pddtab

Displays disk pdd table entries.

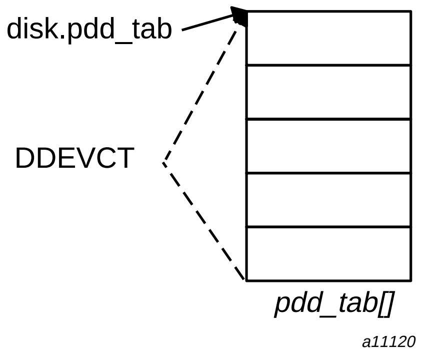

Figure 29. pddtab Structures

3.28 pemap

Displays PE allocator's map entries.

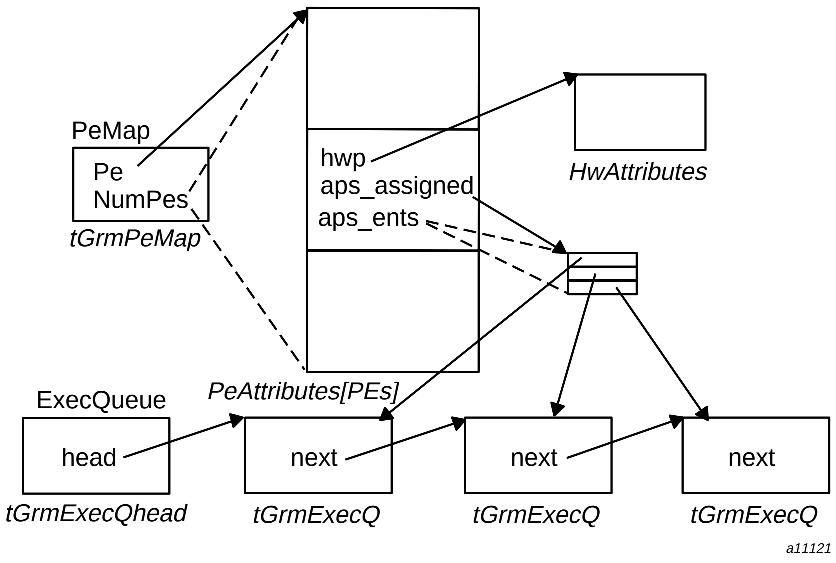

Figure 30. PE Map Structures

3.29 pequeue

Display PE allocator's application queue entries.

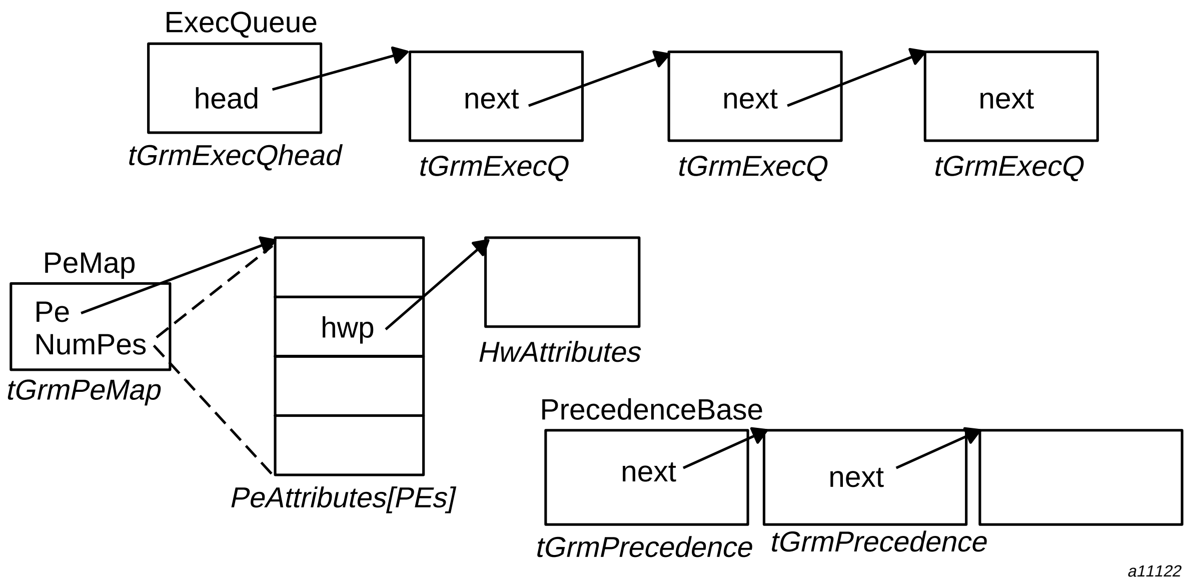

Figure 31. PE Queue Structures

3.30 `pool`

Display system pool structures.

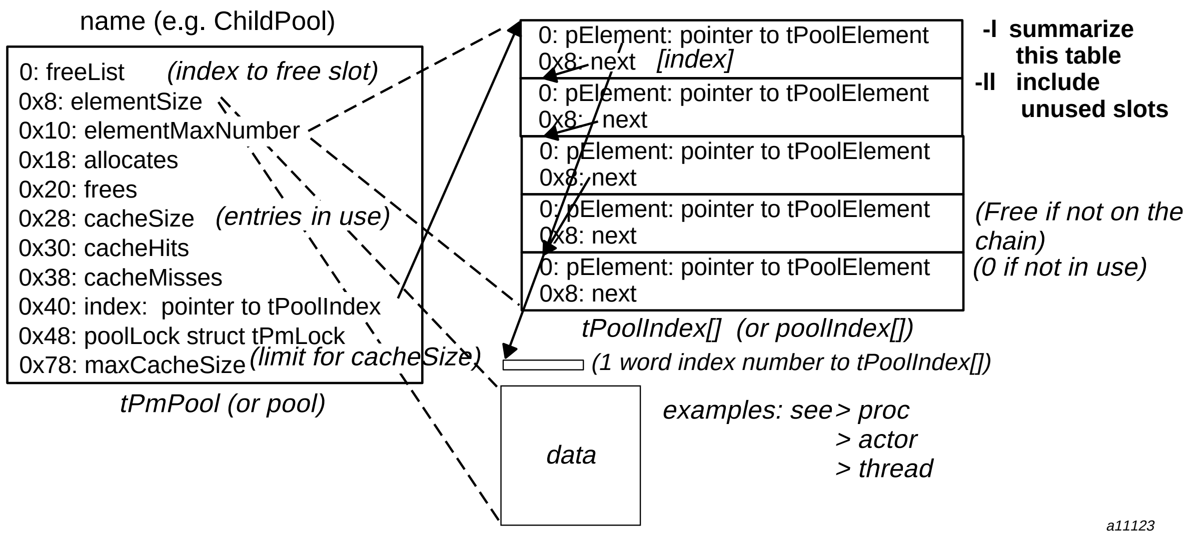
name (e.g. ChildPool)
0: freeList (index to free slot)
0x8: elementSize
0x10: elementMaxNumber
0x18: allocates
0x20: frees
0x28: cacheSize (entries in use)
0x30: cacheHits
0x38: cacheMisses
0x40: index: pointer to tPoolIndex
0x48: poolLock struct tPmLock
0x78: maxCacheSize (limit for cacheSize)
tPmPool (or pool)
0: pElement: pointer to tPoolElement
0x8: next [index]
-l summarize this table
-ll include unused slots
(Free if not on the chain)
(0 if not in use)
tPoolIndex[] (or poolIndex[])
(1 word index number to tPoolIndex[])
data
examples: see > proc
> actor
> thread
a11123

Figure 32. pool Structures

3.31 pput

Validate the physical page usage table (PPut) and display page usage table information.

With extended flow, _KftrLastRt points to kft_info structures, which are defined as follows:

```
struct kft_info {
      uint    kft_tnb0;               /* tnb for caller */
      uint    kft_tnb1;               /* tnb for callee */
      uint    kft_ncalls;             /* number of calls */
      uint    kft_time;               /* Real time clocks */
      struct  kft_info *kft_link;     /* link to next kft_info for tnb */
};
```

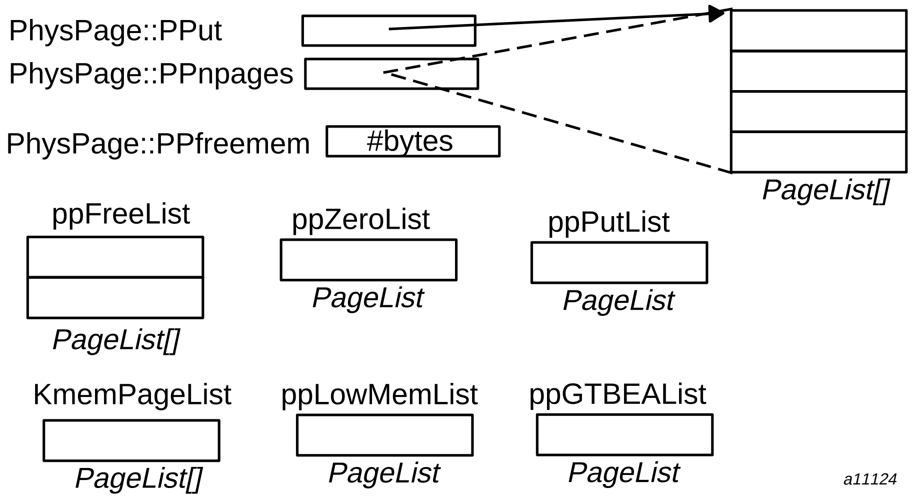

Figure 33. `pput` Structures

3.32 `prnode`

Displays prnode (`/proc rnode`) table entries.

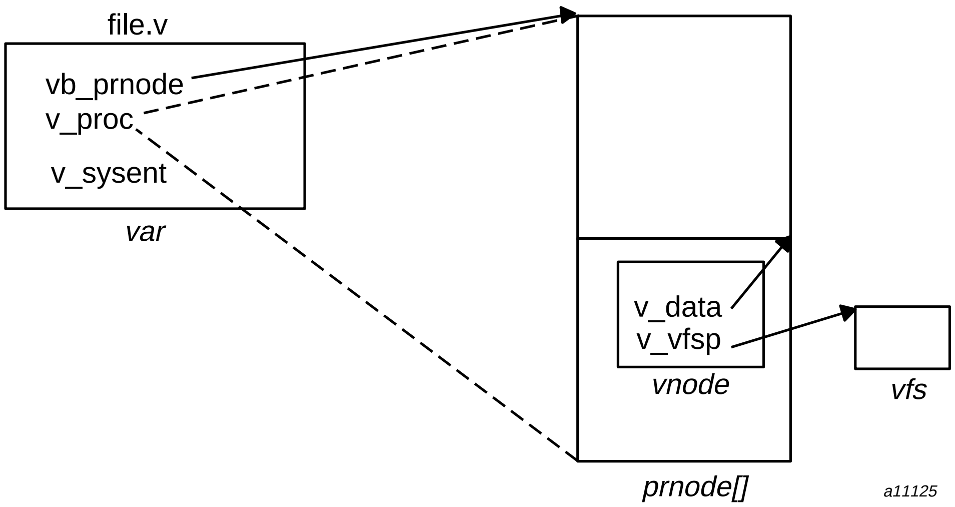

Figure 34. `prnode` Structures

3.33 proc

Displays process information.

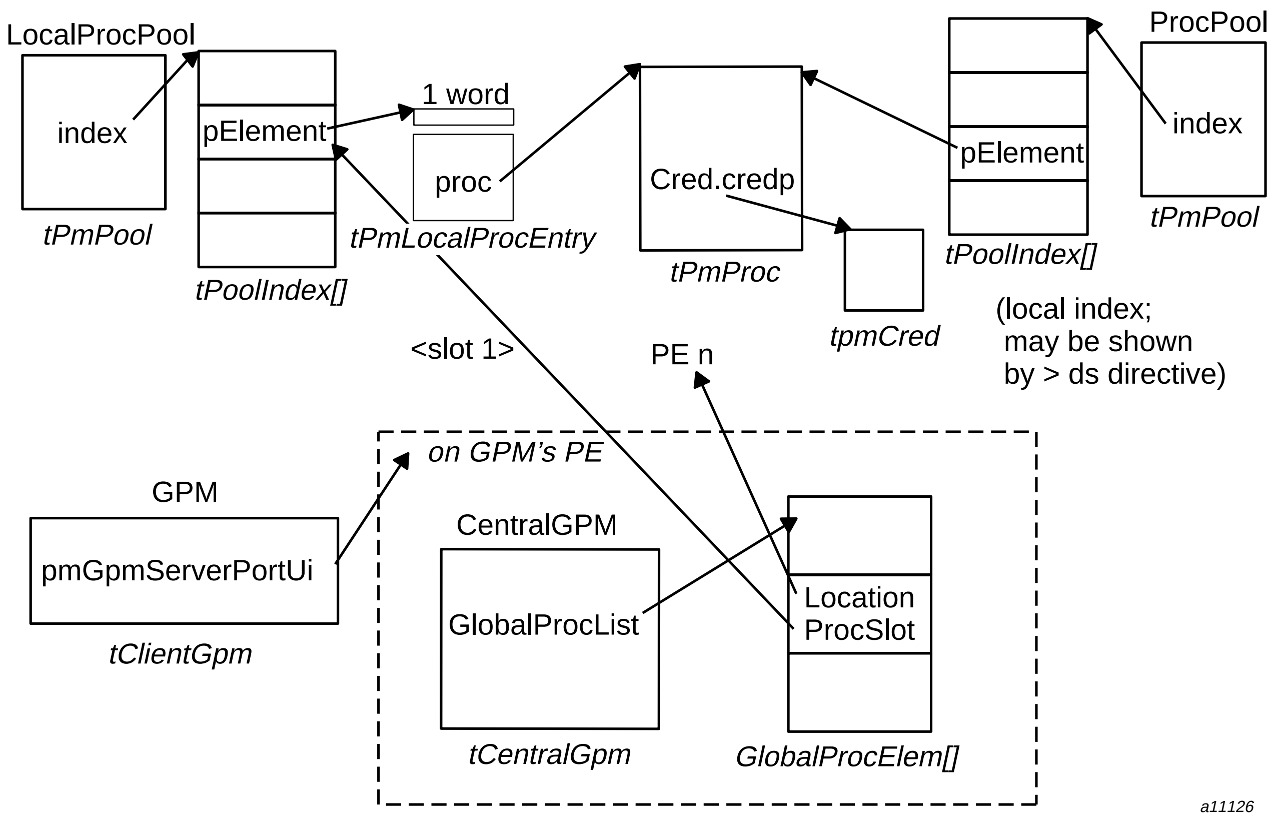

Figure 35. proc Structures

3.34 psched

Displays political scheduler information.

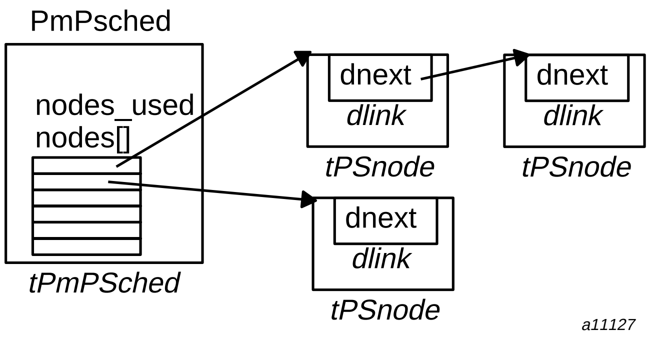

Figure 36. psched Structures

3.35 pty

Displays pty (pseudo tty) table entries.

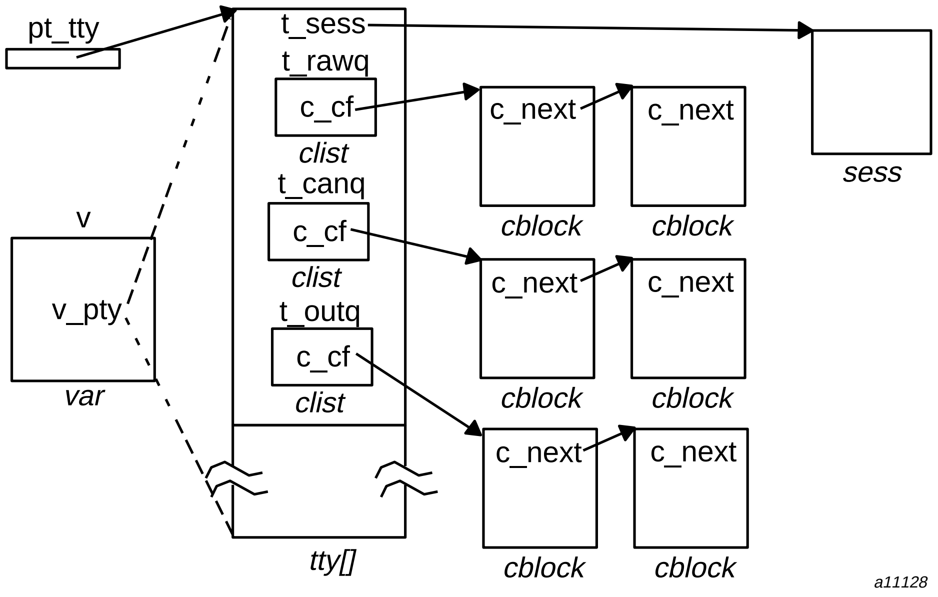

Figure 37. pty Structures

3.36 reg

Displays registers.

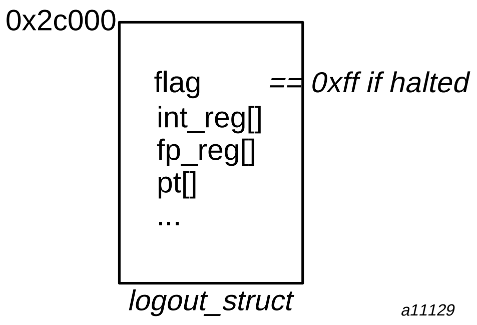

Figure 38. `regs` Structures

3.37 rmon

Displays R-option (network router) performance monitor array information.

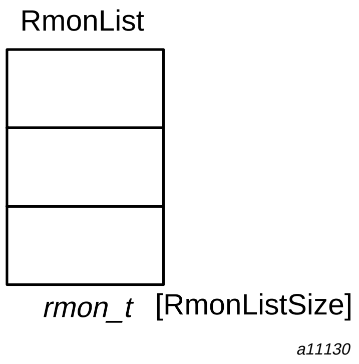

Figure 39. `rmon` Structures

3.38 rnode

Displays `rnode` (NFS remote node) entries.

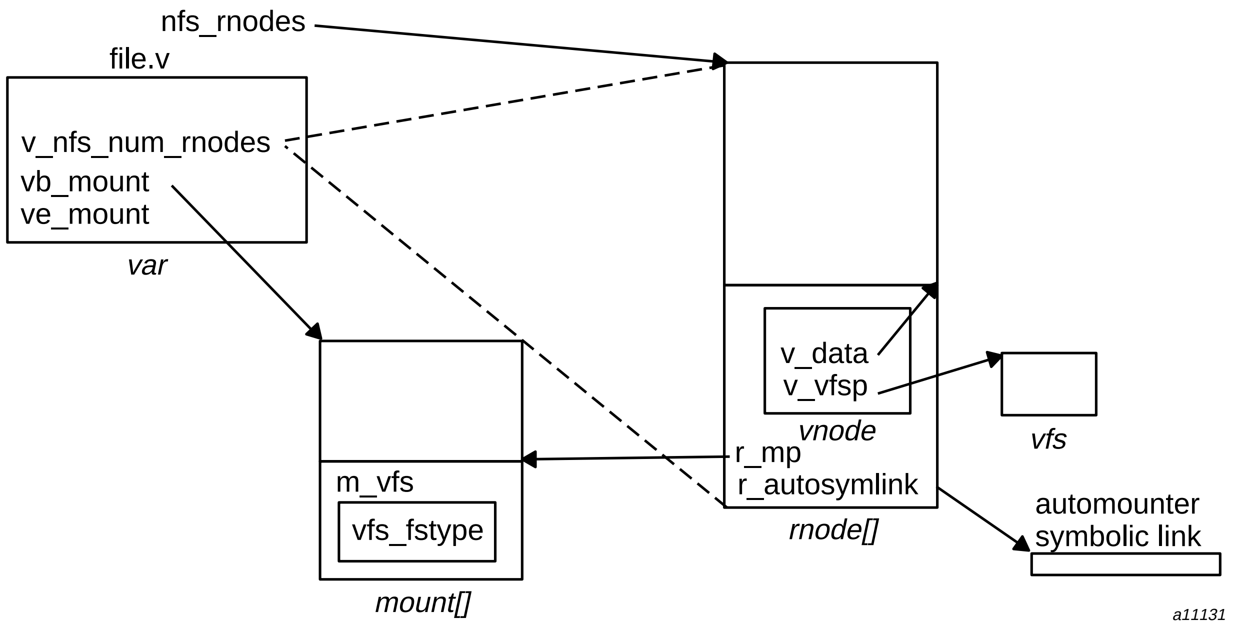

Figure 40. `runq` Structures

3.39 runq

Displays run queue (`runQueue`) entries.

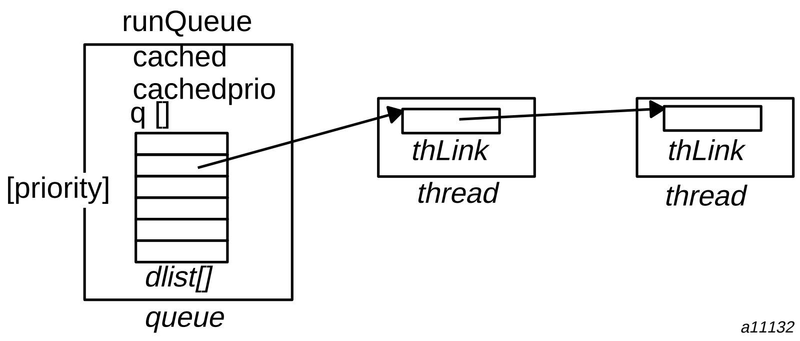

Figure 41. `runq` Structures

3.40 scp

Displays swap_control structures.

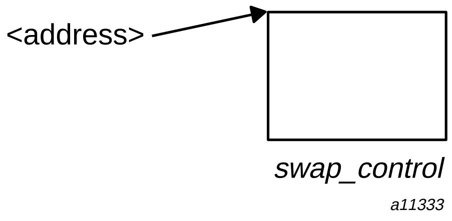

Figure 42. scp Structure

3.41 sem

Displays semaphore information.

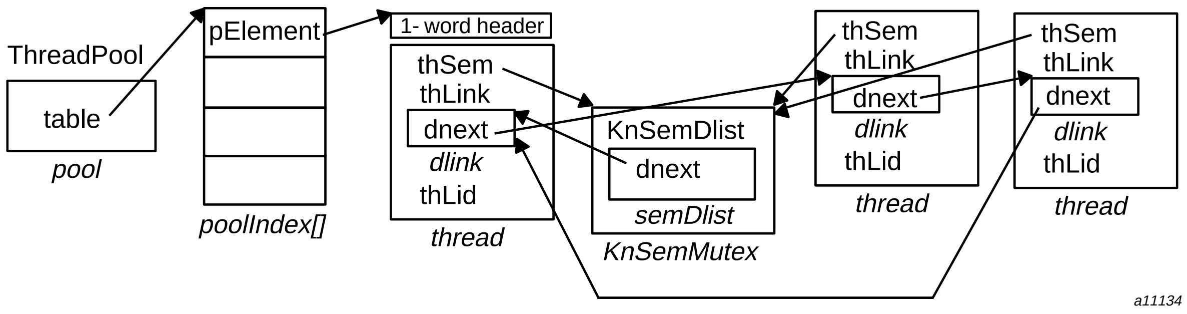

Figure 43. sem Structures

3.42 sess

Displays session table entries.

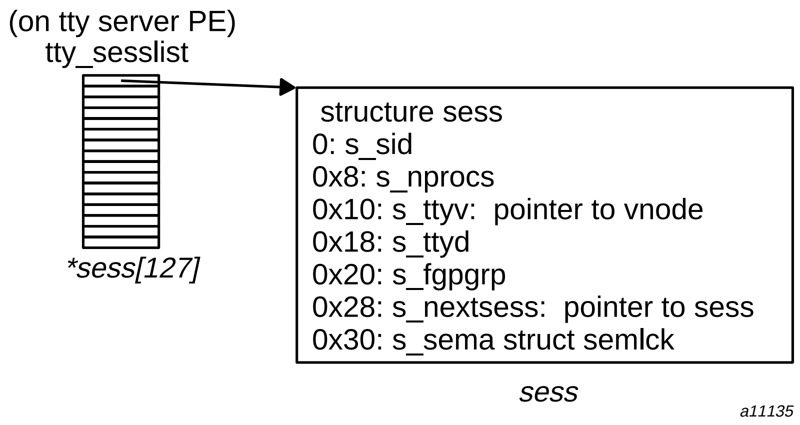

Figure 44. `session` Structures

3.43 `stack`

Displays kernel stack function call traces.

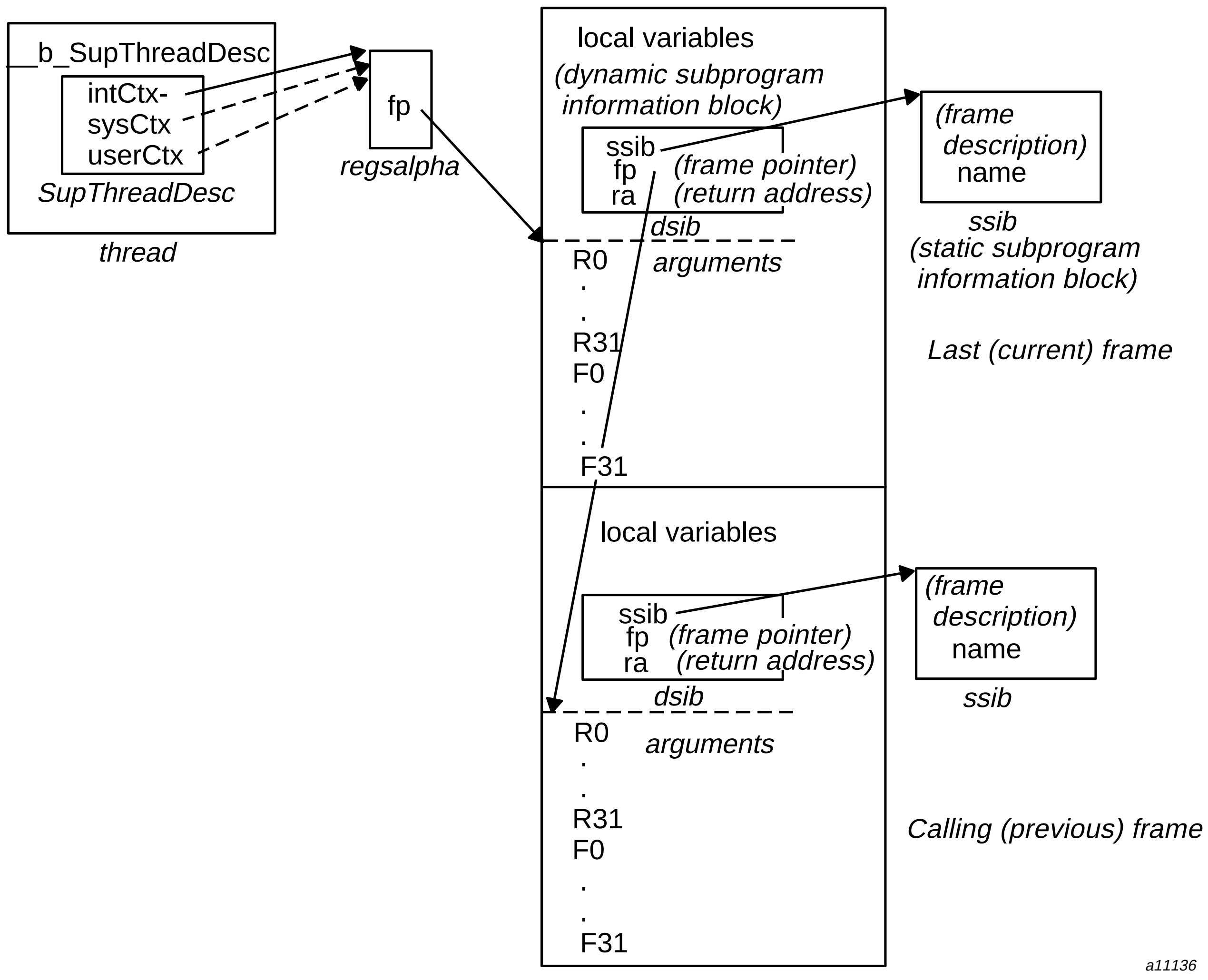

Figure 45. stack Structure

The ssib is defined as follows:

```
struct {
         unsigned     :4;                 // - unused
         unsigned     fpcount:12;         // - # formal args
         unsigned     ty:5;               // - register arg info: reg type
         unsigned     ct:3;               //    # reg args
```

```
        unsigned     ver:8;             // - ssib version ( == 2)
        unsigned     kind:8;            // - kind of subprogram
        unsigned     lang:8;            // - language
        unsigned     len:8;             // - length of subprogram name
        unsigned     length:8;          // - ssib length (excluding name)
        unsigned saved_r : 32;          // - saved R-regs
        unsigned saved_f : 32;          // - saved F-regs
        unsigned long entry_point;      // - subprogram entry addr
        unsigned long lang_info;        // - language specific info
        unsigned long mach_info;        // - machine specific info
        char     name[MAX_NAME_LENGTH]; // - subprogram name: actual length
                                        //   defined by "len" field in header
} ssib;
```

3.44 **`status`**

Displays system status.

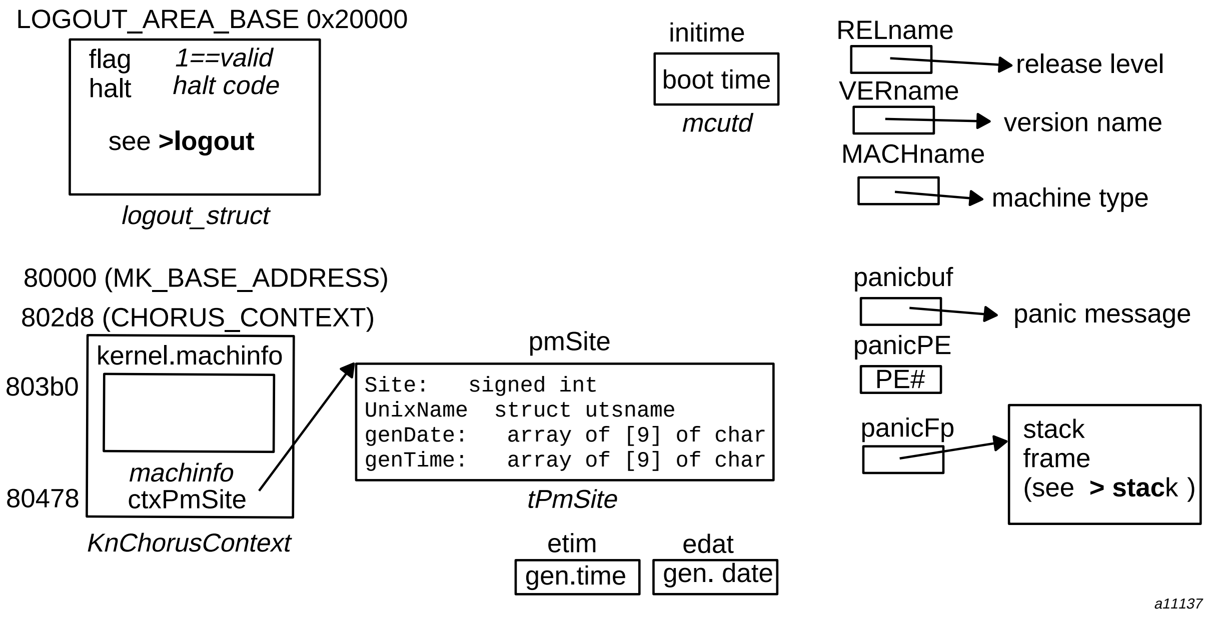

Figure 46. `status` Structures

```
/*-------------------------------------------------------------------
  *                   Unicosmk Context
  *-------------------------------------------------------------------
  */
80000+2d8 = 802d8
typedef struct {
0   VmAddr        ctxUsrStartAddr;
8   VmAddr        ctxUsrEndAddr;
10  VmAddr        ctxSvStartAddr;
18  VmAddr        ctxSvEndAddr;
20  unsigned long ctxLoadTable;
28  unsigned long ctxSystemEntry;    /* address of kernel call table */
30  unsigned long ctxKernelEntry;    /* address of internal kernel call table */
38  unsigned long ctxNucleusEntry;   /* address of internal Nucleus call table*/
40  VmAddr ctxScheduler;
48-58 KnWait      ctxWaitInfo;  3 words
60  unsigned int   ctxMicroData;
68  unsigned long ctxSchedFlags;
70  unsigned long ctxSymbolTable;
78  unsigned long ctxInitTable;
80  unsigned long csim_flag;
88  unsigned long panic_string;
90  unsigned long flow_begin;
98  unsigned long flow_end;
a0  char          *config_file; /* address of configuration file */
a8  unsigned long ctxFreeMem;   /* Kept up to date for GRM */
b0  unsigned long ctxLoadAvg;   /* Kept up to date for GRM */
b8  unsigned long ctxUsrMem;    /* For sysconf(_SC_CRAY_USRMEM) */
c0  unsigned long ctxSysMem;    /* Kept up to date for GRM */
c8-d0  ctxTimeVal        ctxTime;  2 words
d8-198 struct machinfo ctxMachinfo; 25 words (0-0x18) 200 bytes (c8)
1a0 unsigned long ctxPmSite;    /* Ptr to pmSite for dump analysis */
} KnChorusContext;
```

3.45 swapper

Displays memory swap scheduler (swapper) information.

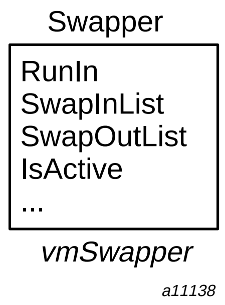

Figure 47. `swapper` Structure

3.46 `swapq`

Displays swap server (`sio`) information.

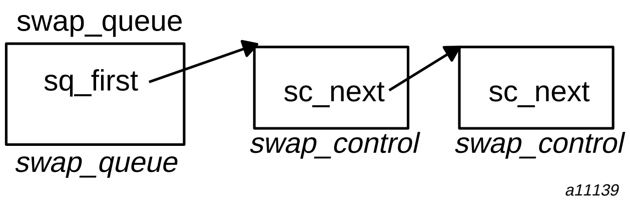

Figure 48. `swapq` Structures

3.47 `syscalls`

Displays system call statistics.

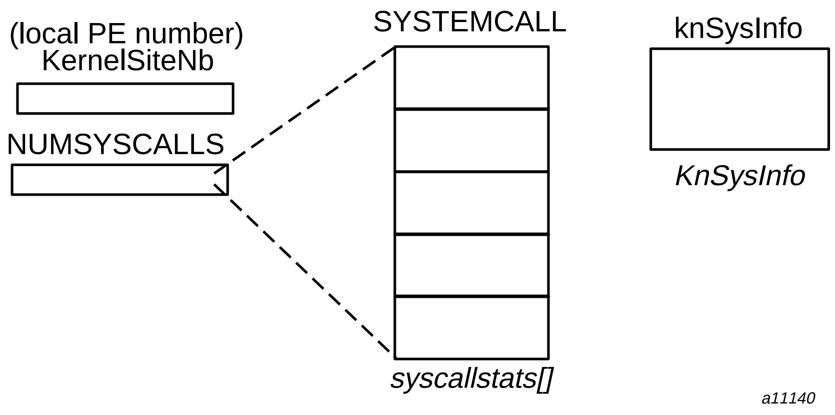

Figure 49. `syscalls` Structures

3.48 `thread`

Displays thread information.

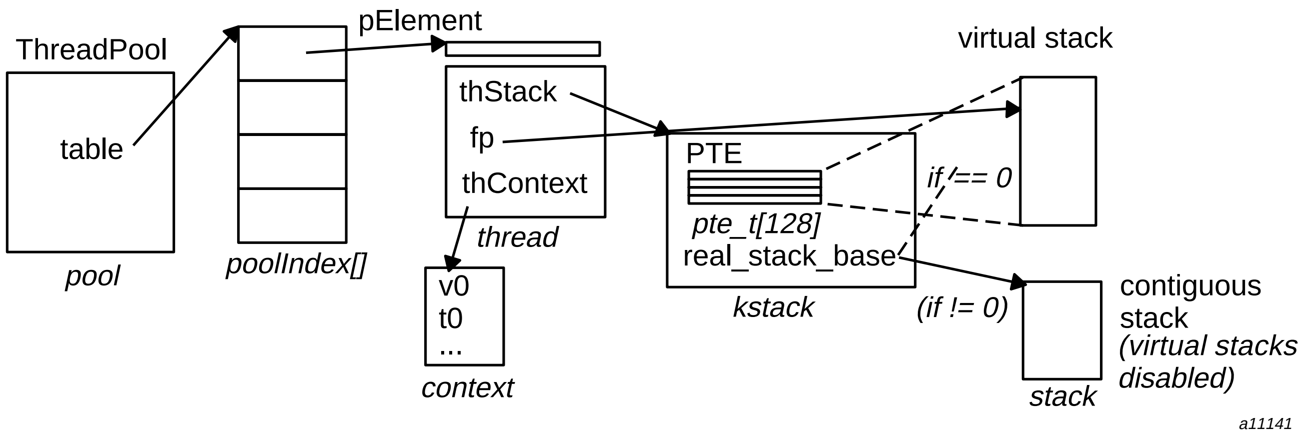

Figure 50. `thread` Structures

3.49 `timeout`

Displays `callout` table entries.

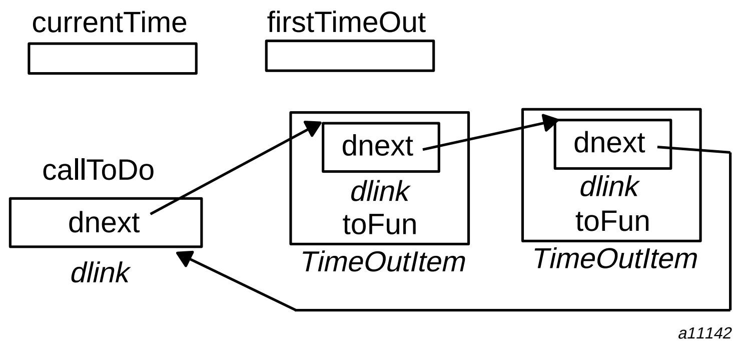

Figure 51. `timeout` Structures

3.50 `tpt`

Displays tape device traces.

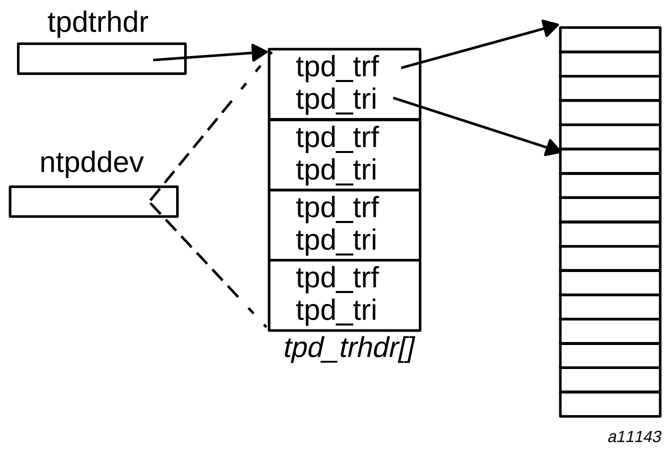

Figure 52. `tpt` Structures

3.51 `traffic`

Displays IPC traffic statistics.

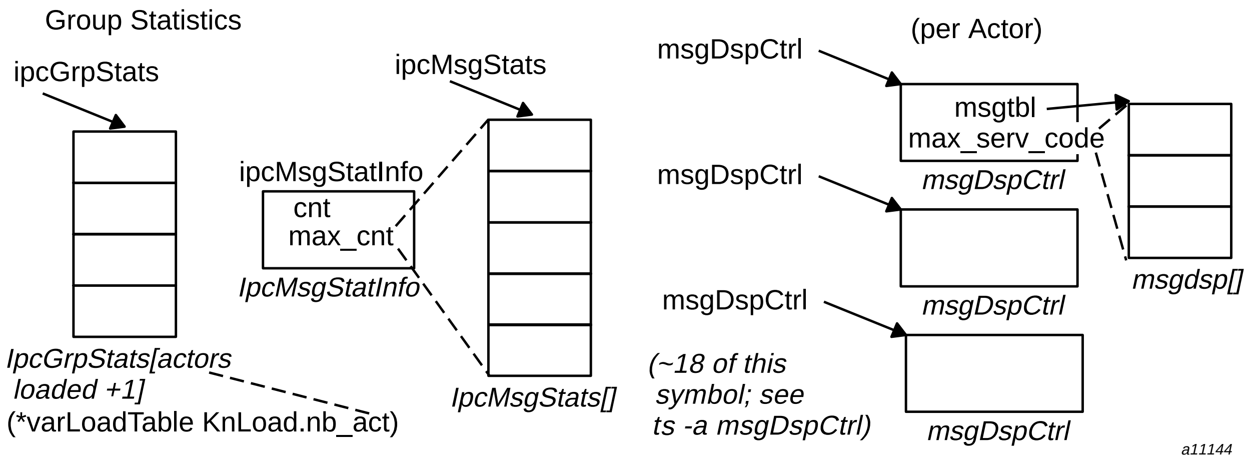

Figure 53. `traffic` Structures

3.52 `tsconv`

Converts a time stamp to date/time, using clock frequency (`mi_hz`) in the `machinfo` structure, and the assumption of 00:00:00 1/1/70 GMT being the beginning of the epoch. If *system_image* is a dump, the time zone is set according to information in the dump file header. This is usually the time zone of the location where the dump was taken. If *system_image* is a live system, the time zone is that of the current location.

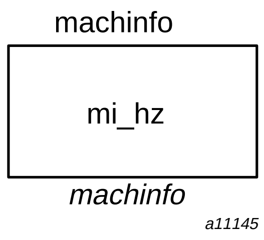

Figure 54. `tsconv` Structures

3.53 `tty`

Displays `tty` table entries.

Also see also the `pty` directive and the `console` directive.

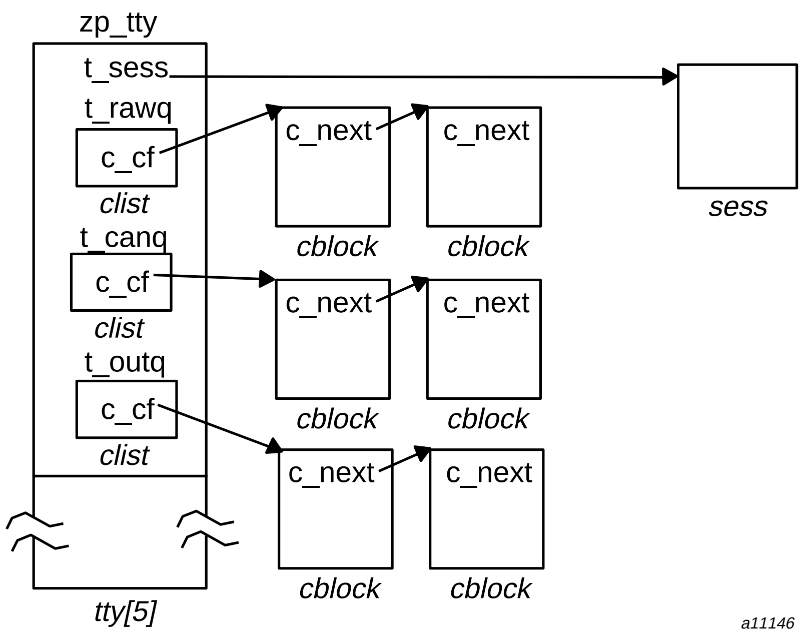

Figure 55. `tty` Structures

3.54 `utrace`

Displays kernel and server trace entries.

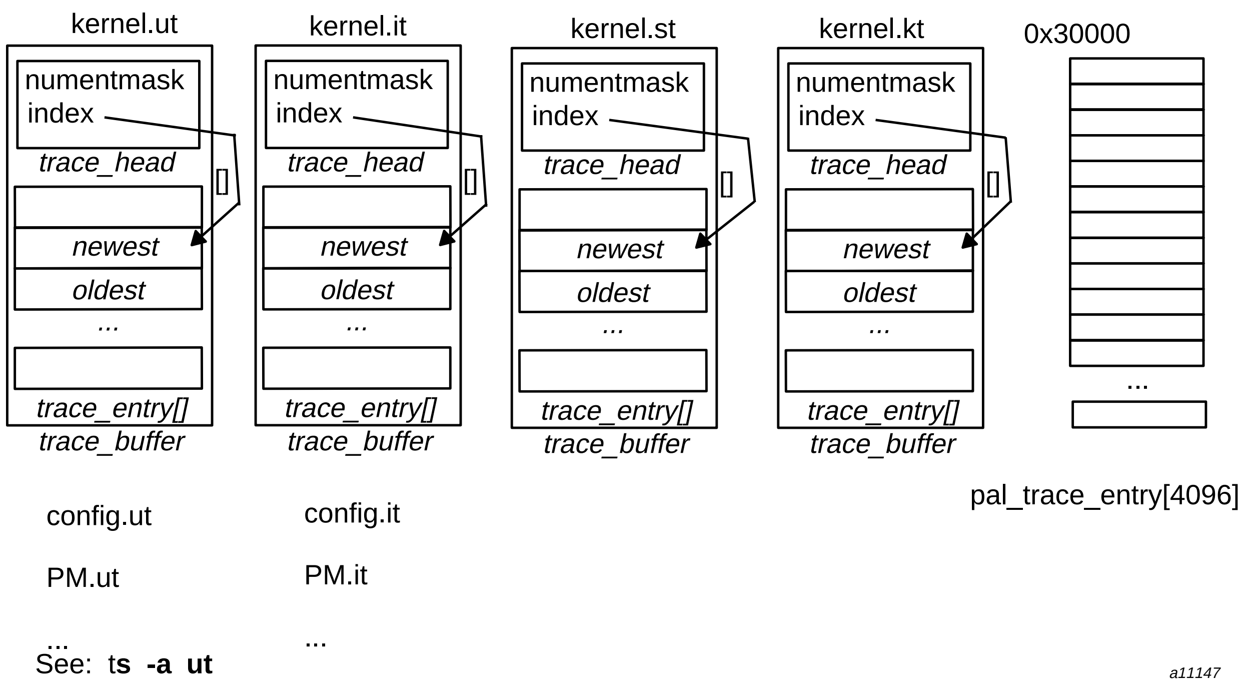

Figure 56. utrace Structures

The latest entry in the PAL trace is determined by scanning all trace time stamps Also see > paltrace. The structure of the PAL trace is defined in crashmk's mkpal.h:

```
/*
 *      MPP PAL Trace Entry
 */
struct pal_trace_entry  {
        union   {
           struct palt_w0a {
               uint    palt_cycle_counter      :32,    /* Cycle counter  */
                       palt_type                :8,    /* Entry type     */
                       palt_pc_addr            :24;    /* PC address     */
           } palt_w0a;
           struct palt_w0b {
               uint    palt_w0;
           } palt_w0b;
       } w0;
       union   {
           struct palt_t0 {            /* Event trace entry (palt_type == 0) */
               uint    palt_exc_addr   :32,    /* EXC_ADDR       */
                                       :5,     /* reserved for MKPAL */
                       palt_fpe        :1,     /* FP enabled */
                                       :22,    /* reserved for MKPAL */
                       palt_user       :1,     /* user mode     */
                       palt_spl        :3;     /* PS(IPL)       */
           } palt_t0;
           struct palt_t1 {           /* Event trace entry (palt_type == 1) */
              uint    palt_reg_val;          /* Register value */
           } palt_t1;
         } w1;
};
```

3.55 var

Displays system variables (var structure).

Example 6: ts -a v

The following actors have var structures:

```
0x000> ts -a v
    PM.v
    fsa.v
    disk.v
    tty.v
    file.v
    socket.v
    GPM.v
```

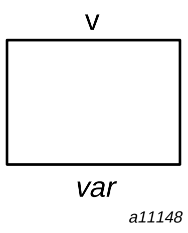

Figure 57. var Structure

3.56 vfs

Displays the mount table in vfs chain order.

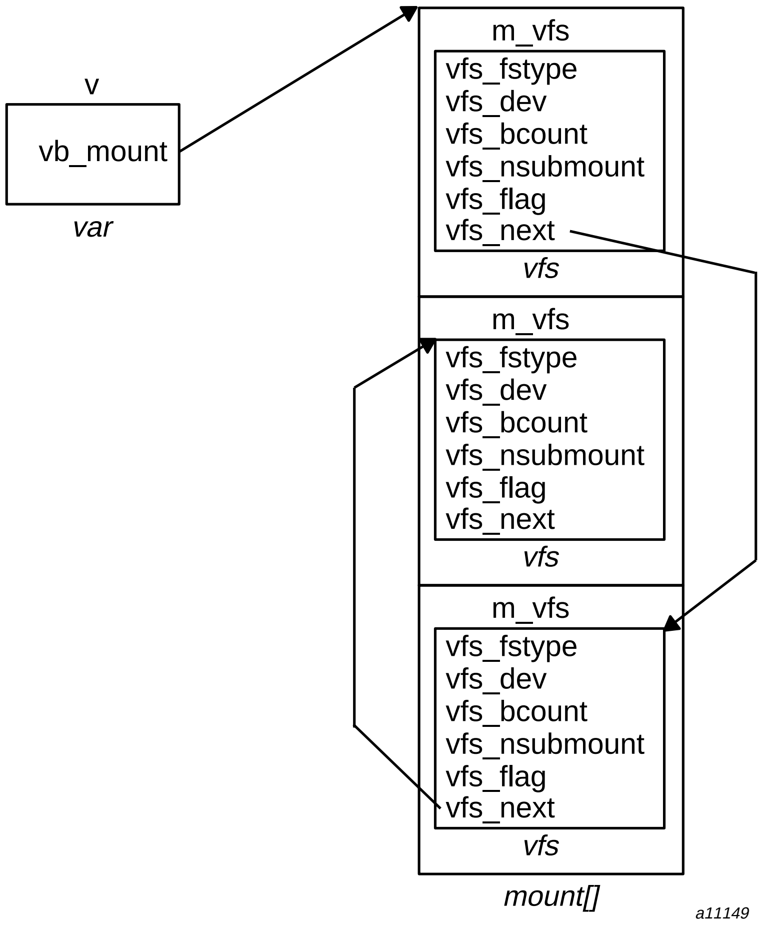

Figure 58. vfs Structures

3.57 vnode

Displays vnodes.

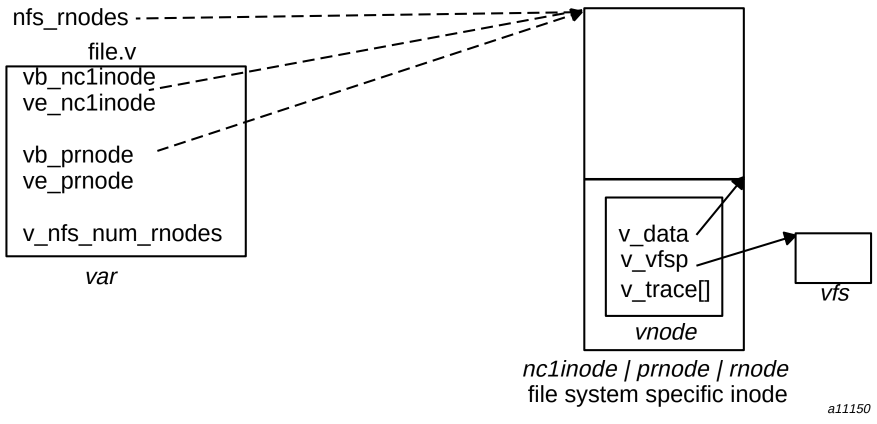

Figure 59. `vnode` Structures

3.58 `xddtab`

Displays `xdd_tab` structures (SCSI disks)

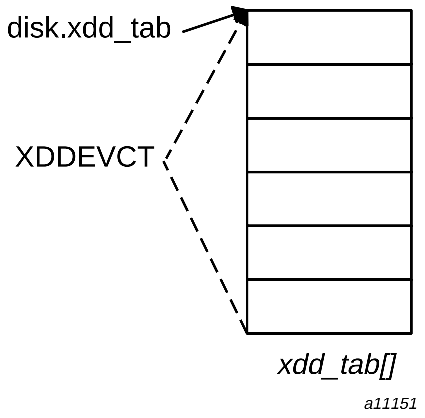

Figure 60. `xddtab` Structures