NAME

csaverify – Checks accounting records for valid data

SYNOPSIS

/usr/lib/acct/csaverify [-n *nqsfile*] [-C *ctmpfile*] [-N *pnqsfile*] [-P *pacctfile*] [-T *tapefile*] [-s *nrec*] [-v] [-D]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The csaverify command checks the specified accounting file(s) for valid data. Messages are written to standard output when data is found that is not valid.

To delete pacct records, use csapacct(8). To delete Network Queuing System (NQS) records, use csaedit(8). You also can remove bad tape data by using csaedit(8).

If you omit file name options, all file types are verified using the defaults in the following list.

File name options are as follows:

-n *nqsfile* Specifies the file name of an NQS accounting file. This is the unprocessed version of *pnqsfile*. The default is /usr/adm/acct/work/Wnqacct.

-C *ctmpfile* Specifies the file name of a connect-time preprocessed accounting file (output from csaline(8)). The default is /usr/adm/acct/work/Pctime.

-N *pnqsfile* Specifies the file name of a preprocessed NQS accounting file (output from csanqs(8)). The default is /usr/adm/acct/work/Pnqacct.

-P *pacctfile* Specifies the file name of a pacct accounting file. The default is /usr/adm/acct/work/Wpacct.

-T *tapefile* Specifies the file name of a tape accounting file. The default is /usr/adm/acct/work/Wtpacct.

Performance options are as follows:

-s *nrec* Specifies that no more than *nrec* invalid records are reported for each file type. The default for *nrec* is 2.

-v Sets verbose mode. Output is written to standard output.

-D Sets debug mode. Output is written to standard output.

BUGS

If a partial accounting record is encountered, the partial record may or may not be flagged as not valid. However, all subsequent records will be noted as being not valid because the header words will be out of alignment.

EXAMPLES

The following example verifies the `Wpacct1 pacct` file with verbose mode turned on.

```
csaverify  -P  Wpacct1  -v
```

FILES

`/usr/include/acct/dacct.h`	Daemon accounting header file
`/usr/include/sys/acct.h`	Accounting records header file
`/usr/include/sys/accthdr.h`	Accounting records header definition file for the `/usr/include/sys/acct.h` file

SEE ALSO

`csaedit`(8), `csaline`(8), `csanqs`(8), `csapacct`(8), `csarun`(8), `csatape`(8)

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

UNICOS Resource Administration, Cray Research publication SG–2302

NAME

cvtmldir – Converts between wildcard and multilevel directory (MLD) structures

SYNOPSIS

```
cvtmldir [-f] [-m] wildcard_path mldir_path
cvtmldir [-f] [-w] mldir_path wildcard_path
```

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The cvtmldir command converts from a wildcard directory to an MLD (-m option), or from MLD to a wildcard directory (-w option). It does this by linking (where possible), or copying files between directory trees. When a copy is required, the security attributes of the copied files are preserved.

When the -m option is specified, cvtmldir creates the multilevel symbolic link named in *mldir_path* and its associated directory. The command then copies the files recursively from *wildcard_path* to *mldir_path*, setting the process label for each file copied. If the full relative path to the file being created under the MLD does not exist, cvtmldir creates the directories in that path at the label of the file.

When the -w option is specified, cvtmldir creates a wildcard directory and copies the directory structure below each labeled subdirectory into the wildcard directory. When a name collision occurs between two files from two different labeled subdirectories, the second file name is made unique, and a warning is printed showing the old name and the new name.

When the -f option is specified, cvtmldir allows you to use the same directory for both *wildcard_path* and *mldir_path*. This allows the conversion of directories in place, and is useful for converting file system root directories.

When converting to a multilevel directory, cvtmldir fails if the directory that is the target of the multilevel symbolic link (*mldir_path*.mld) already exists. When converting to a wildcard directory, cvtmldir fails if the output directory (*wildcard_path*) already exists. The -f option causes cvtmldir to continue processing even if the actual output directory already exists. This is useful for converting from wildcard to MLD and back under the same path name (that is, *wildcard_path* is the same as the symbolic link target of *mldir_path*).

The actual directory that is used as the target of a multilevel symbolic link (*mldir_path*.mld) may exist when the -f flag is specified, but the link itself must not exist. The link is created as part of the conversion when the -m option is specified.

The cvtmldir command accepts the following options:

-f Allows the user to specify the same directory for both *wildcard_path* and *mldir_path*.

-m Converts from a wildcard directory to a multilevel directory.

-w Converts from a multilevel directory to a wildcard directory.

The following example shows the in-place conversion of `/tmp` from a wildcard directory to a MLD:

```
umount /dev/dsk/tmp
mv /tmp /tmp.mld
mount /dev/dsk/tmp /tmp.mld
cvtmldir -f -m /tmp.mld /tmp
spset -l syslow /tmp.mld
```

The following example shows the in-place conversion of `/tmp` from a MLD to a wildcard directory:

```
cd /tmp.mld
ls
cd /
rm /tmp
cvtmldir -f -w /tmp.mld /tmp.mld
umount /tmp.mld
mv /tmp.mld /tmp
mount /tmp
```

NOTES

Because of the way `cvtmldir` resolves path name collisions, it is possible for a different wildcard directory structure to exist after the translation from a wildcard to a MLD and back to a wildcard directory.

EXIT STATUS

The `cvtmldir` command exits with one of the following values:

Value	Description
0	Successful completion.
1	Incorrect command usage.
2	Unable to convert the multilevel directory to a wildcard directory.
3	File name collisions were detected and files were renamed in the process of converting the multilevel directory to a wildcard directory.

SEE ALSO

`mlmkdir`(8), `mlrmdir`(8)

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

General UNICOS System Administration, Cray Research publication SG–2301

NAME

ddms – Provides online disk maintenance capabilities for Cray PVP systems with an IOS model E

SYNOPSIS

/etc/ddms -a disk

/etc/ddms -a info *devicename*

/etc/ddms -a offload -c *cyl* -h *hed* -s *sec* [-e] [-v] *devicename*

/etc/ddms -a restore -c *cyl* -h *hed* [-s *sec range*] [-e] [-v] *devicename*

/etc/ddms -a flaw -c *cyl* -h *hed* -s *sec* [-d *defect address*] [-g *device information*] [-z] [-v] [-u] *devicename* | *devicename-spindle*

/etc/ddms -a unflaw -c *cyl* -h *hed* -s *sec* [-v] [-z] *devicename* | *devicename-spindle*

/etc/ddms -a write -c *cyl range* | -C [-F *flags*] [-h *hed range*] [-S] [-s *sec range*] [-E *errcount*] [-m *mode*] [-p *pattern*] [-b *seed*] [-l *loop*] [-v] *devicename* | [*devicename-spindle*]

/etc/ddms -a read [-c *cyl range* | -C] [-F *flags*] [-h *hed range*] [-S] [-s *sec range*] [-E *errcount*] [-b *seed*] [-p *pattern*] [-l *loop*] [-o] [-v] *devicename* | [*devicename-spindle*]

/etc/ddms -a surf -c *cyl range* | -C [-F *flags*] [-h *hed range*] [-S] [-s *sec range*] [-E *errcount*] [-m *mode*] [-p *pattern*] [-b *seed*] [-l *loop*] [-v] *devicename* | [*devicename-spindle*]

/etc/ddms -a rewrite -c *cyl* -h *hed* -s *sec* [-v] *devicename*

/etc/ddms -a readft [-c *cyl range*] [-f *filename*] [-h *hed range*] [-s *sec range*] [-t *flaw table*] [-u] *devicename* | [*devicename-spindle*]

/etc/ddms -a checkft *devicename*

/etc/ddms -a vid [-c *cyl range* | -C] [-h *hed range*] [-s *sec range*] [-v] *devicename* | [*devicename-spindle*]

/etc/ddms -a format -c *cyl range* | -C [-h *hed range*] [-s *sec range*] [-v] *devicename* | *devicename-spindle*

/etc/ddms -a spmap [-i|-o|-r] [-v] *devicename*

/etc/ddms -a reconstruct [-v] *devicename*

/etc/ddms -a spinup *devicename-spindle*

/etc/ddms -a spindown *devicename-spindle*

/etc/ddms -a scrub [-r] *devicename*

/etc/ddms -a label *devicename*

`/etc/ddms -a ldfrmt` *devicename-spindle*

`/etc/ddms -a backupft` *devicename*

`/etc/ddms -a makeuft -f` *serialnumber devicename*

`/etc/ddms -a restoreft -f` *filename devicename*

`/etc/ddms -a aft` [`-w`] [`-r`] *devicename*

`/etc/ddms -a wrecc -c` *cyl* `-h` *hed* `-s` *sec* [`-b` *bit*] [`-p` *pattern*] [`-l` *length*] [`-m` *mode*] *devicename*

IMPLEMENTATION

Cray PVP systems with an I/O subsystem model E

DESCRIPTION

The `ddms` (disk diagnostic and maintenance system) command provides disk maintenance capabilities by providing for surface testing, flaw maintenance, ID verification, and flaw table integrity capabilities. Errors detected during `ddms` execution are logged into a file named `ddmslog` in your current working directory. You can display errors by using either `olhpa` or `errpt` (for example, `/etc/diag/olhpa -d ddmslog`).

This command requires that you either have `diag` privilege set in the `permbits` field of the user database (UDB), or you are running as super user. In addition, you need `guard` privilege if you plan to run in unguarded mode and write outside the bounds of offloaded sectors (super-user privilege also allows you to run in unguarded mode). See `udbgen`(8).

`-a` Specifies the action to be performed. Valid actions are as follows:

`disk`	Displays valid disk devices and their type as defined in the `/dev/ddd` directory.
`info`	Displays information about the selected disk drives.
`offload`	Assigns a data sector to a spare map sector and moves the data to that sector.
`restore`	Restores a single sector or range of off-loaded sectors.
`flaw`	Updates all flaw tables, reformats the target track/sector, and reassigns unhideable and unslippable sectors to the spare map. Data is preserved for owner access.
`unflaw`	Removes a flaw added with the `flaw` command.
`write`	Writes data patterns and/or random data to the specified `-c`, `-h`, and `-s` range. The `write` action is destructive of data and must be used with caution.
`read`	Allows for the reading of all sectors in the `-c`, `-h`, and `-s` range specified.
`surf`	Writes data patterns to disk, followed by a read-data-and-compare routine. The `surf` action is destructive of data and must be used with caution.

`rewrite`	Rewrites a suspect sector in an attempt to correct a weak write or nonmedia-type ECC error.
`readft`	Reads the flaw table specified by the `-t` option from the disk.
`checkft`	Tests the integrity of all flaw tables for range checks, ascending order, and expected versus actual entries.
`vid`	Verifies sector IDs against what is expected, according to the contents of the User Flaw table on disk.
`format`	Reformats the cylinder, head, and sector range specified, according to the contents of the User Flaw table.
`spmap`	When used with the `-i` option, initializes the spare map by writing the spare map header. The `spmap` action then assigns alternative blocks to all unhideable blocks found in the User Flaw table. When used with the `-o` option, `spmap` reorders the current spare map into sequentially ascending order. The data will be moved to the newly assigned sectors. When used with the `-r` option, `spmap` forces a read of the spare map to kernel memory.
`reconstruct`	Changes from 4-spindle mode to 5-spindle mode. This action is used only with disk arrays. This action should be issued only when a failing spindle in an array has been repaired or replaced. If a spindle failure occurs, use the `pddconf`(8) command to down the failing spindle.
`spinup`	Remotely spins up a spindle on an array that has been swapped out or otherwise spun down. This action must be performed on a spun down disabled spindle before other actions can be done to the spindle.
`spindown`	Remotely spins down a spindle. Typically, this action is used to spin down a disabled spindle from an array prior to removing it.
`scrub`	Writes a pattern across an entire array to set data parity across the array. This action is used only with disk arrays. After the write is complete, the spare map is initialized according to the unhideables from the User Flaw table. The `-r` option is available to read the entire array to check for proper data parity. The `scrub` option does not write to the hardware CE and flaw table cylinders. The `scrub` action is destructive of data and must be used with caution.
`label`	Initializes an array to 5-spindle mode without scrubbing or reconstructing the data. This action is used with disk arrays only. Typically, you should not have to use this action. Instead, you should use the `scrub` action to initialize the array, and use the `reconstruct` action to change an active drive from 4-spindle to 5-spindle mode. The `label` action is destructive of data and must be used with caution.
`ldfrmt`	Loads DCA-2 or DCA-3 device format specifications. This action may need to be done after swapping a spindle from an array.

`backupft` Allows the backing up of the User Flaw table on disk to a file. The file name is dependent on the disk serial number.

`makeuft` Creates a new User Flaw table on the target disk by reading and generating the flaw table information from the sector IDs. This action will overwrite the current User Flaw table.

`restoreft` Allows the restoring to disk of a previously backed-up User Flaw table. The file is identified by the `-f` option.

`aft` Allows you to update or create an `/etc/aft` file (ASCII flaw table file) from the current unhideables table. This file is typically used with the `bb`(8) command to skip bad block assignments. If no option is specified with this action, an ASCII flaw table list is generated and displayed to `stdout`.

`wrecc` Reads the data and ECC on the specified sector, toggles the specified bit(s), and writes the data and ECC. The sector is then read to verify that an ECC error is generated. If the `-p` option is used, the data pattern is written before the ECC is generated on the selected pattern type.

`-b` Supplies a seed value during read and write operations when the `-p rand` option is used. When used with the `wrecc` action, specifies which bit to toggle.

`-c` Specifies a single cylinder or a range of cylinders. The values must be in octal (for example, `-c200-400`, `-c 10`, `-call`).

`-C` Can be used instead of the `-c` option to specify only CE cylinders.

`-d` Used with the `flaw` action to specify the defect address for types of devices that require one.

`-e` Used with the `offload` action to avoid moving the data to the alternative sector.

`-E` Defines the number of errors allowed before `ddms` will abort a `read`, `write`, or `surf` action. The default setting is 500.

`-f` When used with the `-a restoreft` action, supplies the file name for the User Flaw table to be restored. When used with the `-a makeuft` action, supplies the drive serial number. When used with the `-a readft` action, indicates a user flaw file to be displayed, rather than the table on disk.

`-F` *flags* Specifies control flag settings for a specific execution instance. The control flags are defined in the `/usr/include/sys/eslice.h` include file. The default setting is 0137.

`-g` Used with the `flaw` action to specify the head and channel in error. DD-49 type drives require that the general status on an error report be supplied. DD-60 type drives require that a 3-character octal field be specified defining the head(s) containing the defect. This *Combined Head Mask* is supplied with the `olhpa` long report.

`-h` Specifies the desired head or range of heads in octal. (for example, `-h 0-7`, `-h5`, `-hall`).

`-i` Indicates the `spmap` action that initializes the UNICOS spare map.

`-l` Specifies a decimal pass count. When used with the `wrecc` action, specifies the number of bits to be toggled.

`-m` Required to write outside the bounds of off-loaded sectors. The valid options are `guard`, `unguard`, and `destroy`. To validate the `unguard` mode, either you must be running as root or have `guard` privilege set up in the `permbits` field of the user database (UDB). See `udbgen`(8) for more information on setting up permbits. The `destroy` mode works like `unguard` mode, except that it allows writing to the spare map and flaw table cylinders. This option should be used with extreme caution as it will allow customer data to be destroyed.

`-o` Indicates to the `read` action to read in an oscillating fashion. Indicates to the `spmap` action to reorder the spare map sectors into sequentially ascending sectors.

`-p` Specifies the data pattern for a `write`, `surf`, `read`, or `wrecc` action. If `-p` is not specified with the `write` and `surf` actions, `dflt` (all) patterns will be used. If the `-p` option is used with the `read` or `wrecc` action, the data read will be compared against that pattern. Valid options are `zeros`, `ones`, `hilo`, `peak`, `hole`, `bump`, `rand`, `addr`, and `dflt`.

`-r` When used with the `spmap` action, forces a read of the spare map to kernel memory. When used with the `scrub` action, specifies to read the device after the write scrub has completed. When used with `aft` action, reads and displays the contents of the `/etc/aft` file to `stdout`.

`-S` When used with the `read`, `write`, and `surf` actions, specifies to ignore the flaw tables and try to run to unhideable sectors.

`-s` Specifies the sector range desired. Specify the range in octal. Default: all sectors.

`-t` Specifies the flaw table to read in the `readft` action. Valid arguments are `user`, `factory`, `system`, and `unhide`. Default: user.

`-u` If specified with the `readft` action and the `-t user` option, `-u` displays only flaws added by CRI. If specified with the `flaw` action, `-u` indicates an unhideable flaw for defect addressing type devices (for example, a DD-60 drive).

`-v` When specified with most actions, the command output is verbose.

`-w` When used with the `aft` action, writes the `/etc/aft` file with the unhideable entries of the User Flaw table for the specified device.

`-z` Bypasses the request for confirmation on specific actions. By default, default, when a `surf`, `read`, `write`, `vid`, and `format` action are specified, you are required to enter `y` to continue. If the `-z` option is used, this prompt does not appear.

devicename

Specifies the device inode name typically set up in `/dev/ddd`. For disk array type devices (DCA-3), some actions require the specific spindle number to be supplied in addition to the device name. `ddms` checks the *devicename* option for a `-`*spindle*, where *spindle* is the specific spindle number for the request. For example, `0334.1-3` specifies to send the request to spindle 3. See `mknod`(8) for more information on setting up device nodes.

filename Specifies the file created by the `backupft` action.

NOTES

If this command is installed with the default privilege assignment list (PAL), you must have an active `secadm`, `sysadm`, or `diagadm` category to use this command.

EXAMPLES

Example 1: Displays disk information.

```
ddms -ainfo /dev/ddd/0130
```

Example 2a: Flaws out cyl 10 hed 6 sec 22 with channels a1 and a2 of a DD-49.

```
ddms -aflaw -c10 -h6 -s22 -g000140 /dev/ddd/0136
```

Example 2b: Flaws out cyl 10 hed 1 sec 22 as hideable with defect address 321 and head 5 in error for a DD-60.

```
ddms -aflaw -c10 -h1 -s22 -d321 -g040 0136.1
```

Example 2c: Flaws a disk array sector. Notice that the spindle indicator is a required entry.

```
ddms -aflaw -c10 -h1 -s22 -d321 -g040 0136.1-2
```

Example 3: Removes the previously added DD-49 flaw.

```
ddms -aunflaw -c10 -h6 -s22 /dev/ddd/0136
```

Example 4: Reads sequentially all cylinder heads and sectors.

```
ddms -aread 0134
```

Example 5: Reads in an oscillating fashion all cylinders and sectors on head 5.

```
ddms -aread -o -call -h5 -sall 0136
```

Example 6: Reads and compares data to the addressing pattern for all of cylinder 100.

```
ddms -aread -c100 -paddr 0136.4
```

Example 7: Writes all data patterns to cylinders 10 through 20, all heads, and all sectors.

```
ddms -awrite -c10-20 -hall -m unguard 0130
```

Example 8: Initially, writes an addressing pattern to cylinders 100 through 200, all heads, and all sectors. Then the command reads the cylinders, heads, and sectors, and compares data.

```
ddms -asurf -c100-200 -paddr -munguard /dev/ddd/0136
```

Example 9: Shows an alternative to typing in the ddms command and action. You can symbolically link ddms to all the action names preceded by 1 character of your choice, as follows:

```
ddms -aread /dev/ddd/0136
```

could be entered as:

```
aread /dev/ddd/0136
```

if you set up a symbolic link in the following manner:

```
ln -s ddms aread
```

The a character preceding the read command is required so that the read command or any other ddms command will not be mistaken for a UNICOS command or shell built-in command. The a character can be any character value.

SEE ALSO

bb(8) for information on bad block files
mknod(8) for information on setting up device nodes
pddconf(8) for information on downing a failing spindle
udbgen(8) for information on setting up permbits in the user database (UDB)

Online Maintenance Tools Guide for Cray PVP Systems, Cray Research publication SD–1012. (This document contains information private to Cray Research, Inc. It can be distributed to non-CRI personnel only with approval of the appropriate Cray manager.)

NAME

ddoffload – Provides offloading of a single spindle of a DD-4*x* disk drive for Cray PVP systems

SYNOPSIS

```
/etc/diag/ddoffload -s source_device -g good_spindle -F fn
/etc/diag/ddoffload -s source_device -b bad_spindle -g good_spindle [-p]
/etc/diag/ddoffload -s source_device -b bad_spindle -g good_spindle [-f fn] [-p]
/etc/diag/ddoffload -s source_device -b bad_spindle [-d destination_device] -g good_spindle
[-p]
/etc/diag/ddoffload -h
```

IMPLEMENTATION

Cray PVP systems (except CRAY J90 series and CRAY EL series)

DESCRIPTION

The ddoffload (disk-drive offload) command is designed to copy (offload) all data from a spindle that is going bad to a spare spindle through a series of cable swaps and flaw table manipulations. Use ddoffload only when the failing spindle is in a condition to allow data read operations. This command is valid only for DD-40, DD-41, DD-42, and DD-4R device types.

-s *source_device*
: Required. Defines the source device that contains the bad (failing) spindle.

-b *bad_spindle*
: Required except with -F *fn*. Defines the bad (failing) spindle on the source device.

-g *good_spindle*
: Required. Defines the good spindle location on the destination device. This location is used for the data transfer between the bad and spare spindles. Also defines the spindle path to use for the -F *fn* option to create a spare spindle flaws (*fn*) file.

-d *destination_device*
: Required if a two-cabinet offload is being performed. Defines the destination device that contains the good spindle. Only this location is used for the transfer.

-F *fn* Creates a spare flaws file with the specified name. This option only creates a spare flaws file; it does not start the offload process. Performing a ddoffload -s *source_device* -g *good_spindle* -F *fn* in advance saves two cable swaps when you are doing a full offload (one cabinet).

-f *fn* Specifies the existing spare flaws file. Use the specified spare flaws file already created with the -F option. When -f *fn* is not specified, the default file spareflaws is created and used.

-p Disables ddoffload from plocking into memory during the data transfer. By default, ddoffload plocks into memory.

`-h` Displays a help screen for `ddoffload`.

EXAMPLES

Example 1: Creates a file called `spare40`, which contains all unhideable flaws from the spare spindle. You can use the spare spindle later for a full offload. No data offload is performed. To save time when doing an actual offload for a bad spindle, it is recommended that you create a file ahead of time.

```
ddoffload -s 40-A1-27 -g 2 -F spare40
```

Example 2: Performs a data offload from device `40-A1-27`, spindle `3`, to the spare spindle, using the location of spindle `1`. The spare flaws file, `spare40`, which was created in the previous example, is used. Specifying `-p` causes `ddoffload` not to plock into memory during the data transfer and verification portion of the process (one-cabinet method).

```
ddoffload -s 40-A1-27 -b 3 -g 1 -p -f spare40
```

Example 3: Performs a data offload from device `40-A1-27`, spindle `1`, to device `40-A1-26`, spindle `1`. This command saves two cable swaps, because two DD-40 cabinets (channels) are used (two-cabinet method).

```
ddoffload -s 40-A1-27 -b 1 -d 40-A1-26 -g 1
```

Example 4: Performs a data offload from device `/dev/ddd/0334.3`, spindle `2`, to device `/dev/ddd/0335.0`, spindle `0`. This example is for an IOS model E system with two DD-41 drives (two-cabinet method).

```
ddoffload -s 0334.3 -b 2 -d 0335 -g 3
```

SEE ALSO

Disk Drive Offload User Guide (Version 2.0), publication CDM–1028–000. (This manual is Cray Research Proprietary; dissemination of this documentation to non-CRI personnel requires approval from the appropriate vice president and a nondisclosure agreement. Export of technical information in this category may require a Letter of Assurance.)

NAME

`ddstat` – Displays configuration information about disk type character and block special devices

SYNOPSIS

`/etc/ddstat` [`-d`] [`-l`] [`-m`] [`-r`] *special0 special1* ...

IMPLEMENTATION

Cray PVP systems with I/O subsystem model E

CRAY J90 series

CRAY EL series

DESCRIPTION

The `ddstat` command parses specified disk device inodes and formats information about each. Logical devices are divided into their individual components and presented in a disk-specific format that is similar to the output of the `file`(1) command.

The `ddstat` command accepts the following options and operands:

`-d` Displays extensive internal debugging information.

`-l` Formats and displays any intermediate `/dev/ldd` nodes.

`-m` Prints a modified display format. The modified display takes the form of the shell script commands that would, when executed, replicate the displayed device special nodes being displayed. The `-m` option forces `-l` display mode.

`-r` When used with the `-m` (modified display format) option, adds the shell script command line (`/bin/rm`) necessary to remove the device special node prior to recreating it with `/etc/mknod`.

special ... The path to the defined character or block device. Possible paths include the following:

```
/dev/ce/*
/dev/ddd/*
/dev/dsk/*
/dev/ift/*
/dev/mdd/*
/dev/pdd/*
/dev/sdd/*
/dev/ssdd/*
/dev/spare/*
```

The disk device man pages contain the device-specific configuration parameters as presented to the mknod(8) command. See dsk(4), mdd(4), pdd(4), rdd(4), sdd(4), and ssdd(4) for device specific parameter definitions.

EXAMPLES

Example 1: The following examples show the output of a ddstat command:

```
% ddstat /dev/dsk/tmp60_6
/dev/dsk/tmp60_6 b 34/84 /dev/ldd/tmp60_6
        /dev/pdd/scr1230.0 c 32/90 10 01230 107180 12512 00 0 0 0
        /dev/pdd/scr1232.1 c 32/91 10 01232 107180 12512 00 0 1 0
        /dev/pdd/scr0230.4 c 32/94 10 0230 107180 12512 00 0 4 0
        /dev/pdd/scr0232.5 c 32/95 10 0232 107180 12512 00 0 5 0
        /dev/pdd/scr0234.6 c 32/96 10 0234 107180 12512 00 0 6 0
        /dev/pdd/scr0236.7 c 32/97 10 0236 107180 12512 00 0 7 0
```

Example 2: In the following example, scr_1232.1 is a character device. It has a major device number of 32 and a minor device number of 91. It is of disk type 10 (DD-60) on *iopath* 01232 (cluster 1, IOP 2, channel 32). The start address is sector 107180, the length is 12512 sectors, the flags are 00, no *altpath* is defined, and the disk is on unit 1 of the disk channel in the IOP.

```
% ddstat /dev/pdd/scr_1232.1
        /dev/pdd/scr1232.1 c 32/91 10 01232 107180 12512 00 0 1 0
```

SEE ALSO

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

dsk(4), mdd(4), pdd(4), rdd(4), sdd(4), ssdd(4) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

NAME

`ded` – Runs a program in dedicated mode

SYNOPSIS

`/etc/ded` [`-v`] [`-t` *timelimit*] *command* [*arguments*]

IMPLEMENTATION

Cray PVP systems

DESCRIPTION

The `ded` command runs a program in dedicated mode and ensures that only one user is running a dedicated program (with `ded`) at a time. If two or more users attempt to use `ded` at the same time, all but one will wait and a message to that effect is placed on standard error.

The *command* with *arguments* is executed with as many CPUs dedicated to its use as it requires. The *command* is limited to a wall-clock time of 10 seconds. (Sites that are licensed for UNICOS source can change this value by changing the line `#define TIMELIMIT 10` in the file `ded.c`.)

The `ded` command accepts the following options:

`-v`	Sets verbose mode. This option causes `ded` to display extra messages about its actions.
`-t` *timelimit*	Lowers the time limit allowed the *command* by *timelimit* seconds. This function is useful in ensuring that very short commands do not waste time.

NOTES

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

Privilege Text	**Action**
`exec`	Allowed to use this command.

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

Active Category	**Action**
`system`, `secadm`	Allowed to use this command. Allowed to specify any command file.
`sysadm`	Allowed to use this command. Allowed to specify any command file, subject to security label restrictions on the file's path. Shell redirected I/O is subject to security label restrictions.

If the PRIV_SU configuration option is enabled, the super user is allowed to use this command and is allowed to specify any command file.

The default generation procedures do not set the setuid bit for the ded command, which is required for it to work. Thus, sites cannot install it unknowingly. A site that wants to enable this command must execute the following command lines. First, to change the owner of the /etc/ded file to root:

```
chown root /etc/ded
```

Second, to change the permissions mode of the /etc/ded file to make it setuid:

```
chmod 4755 /etc/ded
```

The ded command places the *command* in a separate job before executing it. If job accounting is desired, *command* should be a script that enables job accounting and prints job accounting information before exiting.

The ded command depends on the real-time interface defined by /dev/cpu. Programs run in dedicated mode using ded may not be truly dedicated if other real-time processes exist in the system.

A log of invocations of ded is maintained in /etc/ded_log.

SEE ALSO

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

cpu(4) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

NAME

devacct – Controls device and performance accounting

SYNOPSIS

/usr/lib/acct/devacct [-v] -b *type*

/usr/lib/acct/devacct [-v] -l *type filesystem*

/usr/lib/acct/devacct [-v] -t *type*

/usr/lib/acct/devacct [-v] -L *filesystem*

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The devacct command lets system administrators control device and performance accounting.

The devacct command accepts the following options and arguments:

-v	Enables verbose mode.
-b *type*	Begins accounting for the specified *type*.
-l *type filesystem*	Labels *filesystem* with the specified *type*. Only block special devices must be labeled.
-t *type*	Terminates accounting for the specified *type*.
-L *filesystem*	Lists the type currently associated with the specified *filesystem*.

The *type* argument is the device name associated with one of the BLOCK_DEVICE*x* or CHAR_DEVICE*x* variables or the string associated with PERF_NAME0 in the /etc/config/acct_config file. If the *type* contains shell separators, you must enclose the argument in double quotation marks. The *filesystem* argument is the name of a block special device.

BUGS

Only native Cray Research file systems are supported for device accounting.

EXAMPLES

Example 1: The following example labels a file system with type dd40 with ldcache, and then enables accounting for this type:

```
/usr/lib/acct/devacct -l "dd40with ldcache" /dev/dsk/mydd40
/usr/lib/acct/devacct -b "dd40with ldcache"
```

Example 2: The following example disables performance accounting:

```
/usr/lib/acct/devacct -t perf_01
```

Example 3: The following example enables accounting for the Hardware Performance Monitor (HPM):

```
/usr/lib/acct/devacct -b hpm
```

FILES

`/etc/config/acct_config` Accounting configuration file

SEE ALSO

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

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

UNICOS Resource Administration, Cray Research publication SG–2302

NAME

devnm – Prints device name

SYNOPSIS

/etc/devnm *path*

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The devnm command identifies the special file associated with the mounted file system in which the operand *path* resides. As a special case, when *path* is "/" (the root file system), both the block device name and the swap device name are printed for the / argument name if swapping is performed on the same disk section as the root file system. Argument names must be full path names.

The devnm command is most commonly used by /etc/rc (see brc(8)) to construct a mount table entry for the root device.

NOTES

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

Active Category	Action
system, secadm	Allowed to specify any path.
sysadm	Allowed to specify any path, subject to security label restrictions. Shell redirected I/O is subject to security label restrictions.

If the PRIV_SU configuration option is enabled, the super user is allowed to specify any file system.

EXAMPLES

If /usr is the mount point for /dev/dsk/usr_sd, the following command line:

```
/etc/devnm /usr
```

produces the following:

```
/dev/dsk/usr_sd /usr
```

FILES

/dev/dsk/*

SEE ALSO

brc(8)

NAME

dgdemon – Invokes the diagnostic daemon

SYNOPSIS

/etc/dgdemon

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The diagnostic daemon, dgdemon, is used by the System Maintenance and Remote Testing Environment (SMARTE) to execute all online diagnostic tests and concurrent maintenance utilities from SMARTE. It sends all diagnostic output to the maintenance workstation (MWS) or operator workstation (OWS). dgdemon also starts the SMARTE threshold server, thserver, to get real-time notification of error records from the operating system through a socket connection to the error-logging daemon, errdemon(8).

The server collects the records received from errdemon and thresholds errors. Exceeded thresholds are communicated to SMARTE on the MWS or OWS.

Typically, dgdemon is started by the rc script at boot time. dgdemon can be terminated by dgstop(8). Only a super user or a privileged user can start dgdemon, and only one diagnostic daemon can be active at any time on an MWS or OWS port. If errdemon is stopped, the server will reconnect to errdemon when errdemon is restarted.

The dgdemon daemon invokes the centralized identification and authorization library routines to validate the user ID and password.

NOTES

If this command is installed with the default privilege assignment list (PAL), you must have an active secadm, sysadm, or diagadm category to use this command.

MESSAGES

dgdemon: Event processing failed
: dgdemon encountered a bad file descriptor while performing a select(2) system call. dgdemon tried to remove the bad file descriptor. If the file descriptor continues to cause an error, dgdemon will perform an exit(2). To restart dgdemon, execute dgdemon as root. If the problem persists, contact your system support staff.

dgdemon: Cannot get host name
: The gethostname(2) system call failed. dgdemon performed an exit. Contact your system support staff.

`dgdemon: Diagnostic daemon already started on` *host_name*
A copy of `dgdemon` was already running on *host_name*. `dgdemon` performed an `exit`.

`dgdemon: Cannot export TCP/IP connection`
`dgdemon` was unable to export its TCP/IP connection. `dgdemon` performed an `exit`. Ensure that SMARTE is executing on the MWS or OWS. If SMARTE is currently executing on the MWS or OWS, execute `dgstop`, and ensure that `dgdemon` is no longer executing. Then execute `dgdemon` as `root`. If the problem persists, contact your system support staff.

`dgdemon: Cannot connect to the Log Manager`
`dgdemon` was unable to connect to SMARTE on the MWS or OWS. `dgdemon` did not perform an `exit`. When SMARTE is started on the MWS or OWS, `dgdemon` automatically connects to it. This problem does not cause `dgdemon` to exit. Contact your system support staff to start SMARTE on the MWS or OWS.

SEE ALSO

`dgstop`(8) for information on terminating the diagnostic daemon
`errdemon`(8) for information on invoking the error-logging daemon

`ia_failure`(3C), `ia_mlsuser`(3C), `ia_success`(3C), `ia_user`(3C) for information on centralized identification and authorization in the *UNICOS System Libraries Reference Manual*, Cray Research publication SR–2080

`err`(4) for information on the error-logging interface in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

System Maintenance and Remote Testing Environment (SMARTE) Guide, Cray Research publication SD–1017. (This document contains information private to Cray Research, Inc. It can be distributed to non-CRI personnel only with approval of the appropriate Cray manager.)

NAME

dgstop – Terminates the diagnostic daemon

SYNOPSIS

/etc/dgstop

IMPLEMENTATION

Cray PVP systems

DESCRIPTION

The dgstop command sends a software termination signal (SIGTERM) to the diagnostic daemon, dgdemon(8). Only a super user can terminate the diagnostic daemon.

Do not terminate dgdemon independently from errdemon(8). To terminate dgdemon, terminate errdemon by using errstop(8).

SEE ALSO

dgdemon(8) for information on invoking the diagnostic daemon
errdemon(8) for information on invoking the error-logging daemon
errstop(8) for information on terminating the error-logging daemon

NAME

diagccmt – Online I/O diagnostic client

SYNOPSIS

All GigaRing based Cray Research systems:

/etc/diag/diagccmt [-h] [-t]

Cray system workstation (SWS):

/opt/CYRIccmt/bin/diagccmt [-h] [-t]

IMPLEMENTATION

All GigaRing based Cray Research systems
Cray system workstation (SWS)

DESCRIPTION

The diagccmt program is the diagnostic client for the online I/O exercisers.

When you connect to diagccmt through the xdi(8) graphical interface, a list of I/O exercisers is displayed. Choosing one of these exercisers then allows you to change desired parameters and execute the exerciser. The exerciser executes on the machine on which diagccmt is executing.

The diagccmt command accepts the following options in any order:

-h Specifies that the diagccmt command-line usage information is to be printed to standard output. Once this information is displayed, diagccmt exits.

-t Specifies that a trace file is to be created containing a variety of information reflecting the flow of execution.

NOTES

You must be in the craydiag group to execute this command.

The path to this command and the appropriate environment variable settings are provided by the craydiag and crayadm modules.

ENVIRONMENT VARIABLES

PATH When searching for available exercisers, the PATH environment variable is scanned.

FILES

`/tmp/diagccmt.stdout`	Contains the standard output text from the `diagccmt` daemon
`/tmp/diagccmt.stderr`	Contains the standard error text from the `diagccmt` daemon
`/tmp/diagccmt`	Directory created by `diagccmt` that becomes its working directory
`/tmp/diagccmt/tracefile.*`	Contains tracing information

SEE ALSO

Concurrent Maintenance Tools User's Guide, publication SD–2627
SWS-ION Administration and Operations Guide, publication SG–2204

`ddms`(8)
`vht`(8)
`vst`(8)
`vtt`(8)
`xdms`(8)
`xdi`(8)

NAME

dig – Sends domain name query packets to name servers

SYNOPSIS

/usr/ucb/dig [*@server_name*] [*domain_name*] [*dns_type*] [*dns_class*] [*%ignored-comment*] [*-dig_option*] [*+query_option*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The domain name system (DNS), currently used by a majority of the Internet's component networks, is a large and complex autonomously administrated distributed database that provides a network wide name service. Its size and the autonomously administered nature of the system make it an ideal breeding ground for misconfiguration and other problems.

The dig (domain information groper) command is a flexible command-line tool, which can be used to gather information from the DNS servers. dig has two modes: simple command-line mode, which makes a single query; and batch mode, which produces a query for a list of several queries. All query options are accessible from the command line.

The usual simple use of dig takes the form:

dig *@server_name domain_name dns_type dns_class*

The elements on the command line have the following meanings:

server_name Specifies the server name. May be either a domain name or a dot-notation Internet address. If you omit this optional field, dig attempts to use the default name server for your machine.

Note: If a domain name is named(8), this is resolved by using the domain name system resolver. If your system does not use the DNS system for host name lookup, you may have to specify a dot-notation address. Alternatively, if a server is at your disposal, all that is required is that the /etc/resolv.conf file be present and that it indicate where the default name servers reside, so that *server_name* itself can be resolved. See the resolv.conf(5) man page for information on the /etc/resolv.conf file.

Warning: Changing /etc/resolv.conf affects the standard resolver library and potentially, several programs that use it. As an option, you can set the LOCALRES environment variable to specify a file that is to be used instead of /etc/resolv.conf (LOCALRES is specific to the dig resolver and is not referenced by the standard resolver). If the LOCALRES variable is not set, or the file is not readable, /etc/resolv.conf is used.

domain_name Specifies the domain for which you are requesting information.

dns_type Specifies the type of information (DNS query type) that you are requesting. If you omit *dns_type*, the default is `a`. The following list shows some values for *dns_type*; see RFC 1035 for a complete list.

Value	Description
`a`	T_A (network address)
`any`	T_ANY (all/any information about specified domain)
`mx`	T_MX (mail exchanger for the domain)
`ns`	T_NS (name servers)
`soa`	T_SOA (zone of authority record)
`hinfo`	T_HINFO (host information)
`txt`	T_TXT (arbitrary number of strings)

dns_class Specifies the network class requested in the query. If you omit *dns_clas*, the default is `in`. The following list shows some values for *dns_class*; see RFC 1035 for a complete list.

Value	Description
`in`	C_IN (Internet class domain)
`any`	C_ANY (All/any class information)

Note: The `any` value can specify a class and/or a type of query. `dig` parses the first occurrence of `any` to mean *dns_type* = T_ANY. To specify *dns_class* = C_ANY, you must either specify `any` twice, or set the *dns_class* value by using `-c` option of the `-`*dig_option* value.

%ignored-comment

Includes an argument that is not parsed. This may be useful if you are running `dig` in batch mode. Instead of resolving every *@server_name* in a list of queries, you can avoid the overhead of doing so, and still have the domain name on the command line as a reference. Example:

```
dig @128.9.0.32 %venera.isi.edu mx isi.edu
```

-dig_option Specifies an option that affects the operation of `dig`. The following options are currently available:

`-f` *XXX* Specifies the batch mode file. File *XXX* contains a list of query specifications (`dig` command lines) that will be executed in sequence. Lines that begin with '`;`', '#', or '`\n`' are ignored. Other options can still appear on the command line and are in effect for each batch query.

`-T` *XXX* Specifies time (in seconds) between start of successive queries in batch mode. Can be used to synchronize two or more batch `dig` commands. Default is 0.

`-p` *XXX* Specifies port number. Queries a name server that is listening to a nonstandard port number.

`-P`[*ping_string*] Makes a call to the shell to execute a `ping`(8) command for response time comparison after the query returns. The following `ping`(8) command prints the last three lines of statistics:

```
ping -s myserver 56 3
```

If the optional *ping_string* is present, it replaces `ping -s` in the shell command. You must specify a server name on the command line.

`-t` *XXX* Specifies type of query. Can specify either an integer value to be included in the *dns_type* field or use an abbreviated mnemonic value as listed in the *dns_type* description.

`-c` *XXX* Specifies class of query. Can specify either an integer value to be included in the *dns_class* field or use the abbreviated mnemonic value as listed in the *dns_class* description.

`-envsav` Specifies default environment file. This flag specifies that the environment after all of the arguments are parsed should be saved to a file to become the default environment. Useful if you do not like the standard set of defaults and do not desire to include such a large number of options each time `dig` is used. The environment consists of resolver state variable flags, timeout, retries and the flags that detail `dig` output (see +*query_option*). If you set the `LOCALDEF` environment variable to the name of a file, this is the location to which the default environment is saved. If not, the file `DiG.env` is created in the current working directory.

Note: `LOCALDEF` is specific to the `dig` resolver, and it does not affect operation of the standard resolver library.

Each time `dig` is executed, a check is made for `LOCALDEF`. If it is defined and the file is readable, the environment is restored before any arguments are parsed.

`-[no]stick` Specifies restoration of the default environment. This flag is useful only in batch queries. It indicates whether to restore the default environment (see `-envsav`) before parsing and sending each query. The default environment includes the standard defaults, those from `LOCALDEF`, or those set by `-envset`. The default is `-nostick`.

`-envset` Sets default environment. You can use this flag, in conjunction with `-stick`, for batch query runs. When `-envset` is specified, the environment after the arguments are parsed becomes the default environment for the duration of the batch file.

+*query_option* Specifies change to query packet or `dig` output specifics. `+` is used to specify an option to be changed in the query packet or to change `dig` output specifics. Many of these are the same options accepted by `nslookup`(1). If an option requires an argument, the form is as follows:

+*keyword*[=*value*]

You can abbreviate most keywords. Parsing of the `+` options is very simplistic. A value must not be separated from its keyword by white space. The following keywords currently are available. In column 1, the keyword in parentheses is an abbreviation that you may use (for example, (`deb`)). In column 2, the value in brackets is the default value for that keyword (for example, [`4`]).

Valid Keywords	**Meaning**
[`no`]`debug` (`deb`)	Turns on/off debugging mode [`deb`]
[`no`]`d2`	Turns on/off extra debugging mode [`nod2`]
[`no`]`recurse` (`rec`)	Uses/does not use recursive lookup [`rec`]
`retry=`*retries* (`ret`)	Sets number of retries to *retries* [`4`]
`time=`*sec* (`ti`)	Sets time-out length to *sec* seconds [`4`]
[`no`]`ko`	Keeps/does not keep open option (implies `vc`) [`noko`]
[`no`]`vc`	Uses/does not use virtual circuit [`novc`]
[`no`]`defname` (`def`)	Uses/does not use default domain name [`def`]
[`no`]`search` (`sea`)	Uses/does not use domain search list [`sea`]
`domain=`*name* (`do`)	Sets default domain name to *name*
[`no`]`ignore` (`i`)	Ignores/does not ignore truncation errors [`noi`]
[`no`]`primary` (`pr`)	Uses/does not use primary server [`nopr`]
[`no`]`aaonly` (`aa`)	Uses/does not use authoritative query only flag [`noaa`]
[`no`]`sort` (`sor`)	Sorts/does not sort resource records [`nosor`]
[`no`]`cmd`	Echoes/does not echo parsed arguments [`cmd`]
[`no`]`stats` (`st`)	Prints/does not print query statistics (for example, RTT) [`st`]
[`no`]`qr`	Prints/does not print outgoing query [`noqr`]
[`no`]`reply` (`rep`)	Prints/does not print reply [`rep`]

[no]header (he)	Prints/does not print certain parts of header [he]
[no]Header (H)	Prints/does not print all/any of header [H]
[no]ttlid (tt)	Prints/does not print TTLs and pkt IDs [tt]
[no]ques (qu)	Prints/does not print question section [qu]
[no]answer (an)	Prints/does not print answer section [an]
[no]author (au)	Prints/does not print authoritative section [au]
[no]addit (ad)	Prints/does not print additional section [ad]
pfdef	Sets print flags to default
pfmia	Sets print flags to minimal default
pfset=*value*	Sets print flags to *value*
pfand=*value*	Bitwise and print flags with *value*
pfor=*value*	Bitwise or print flags with *value*

The following pseudo-code example summarizes the retransmission strategy (retry/time) used by the resolver library (see resolver(3C)) when sending datagram queries.

Note: When you use the dig command, the number of servers and/or addresses is always 1.

```
for i = 0 to retry-1
      for j = 1 to num_servers
            send_query
            wait((time * (2**i)) / num_servers)
      rof
rof
```

The keywords pfset, pfand, and pfor make manipulation of the various print options less tedious. The currently defined values are as follows:

PRF_STATS	0x0001	RTT, query host, server host information
PRF_CMD	0x0008	dig command-line echo
PRF_QUES	0x0010	Questions section
PRF_ANS	0x0020	Answers section
PRF_AUTH	0x0040	Authoritative section
PRF_ADD	0x0080	Additional records section
PRF_HEAD2	0x0200	Header flags, section RR counts
PRF_TTLID	0x0400	ttl and packet ID number
PRF_HEADX	0x0700	Any/all packet header information
PRF_QUERY	0x1000	Outgoing query packet information
PRF_REPLY	0x2000	Reply packet information
PRF_SORT	0x8000	Sort various response sections
PRF_DEF	0x27f9	Default dig settings

PRF_MIN	0xa133	Minimalistic dig settings for (future) automated server testing

If you want to see information other than statistics when you are setting the print options, you should examine the outgoing packet (0x1000), incoming packet (0x2000), or both packets, plus the specific sections of the packet in which you are interested.

NOTES

The dig command uses a slightly modified version of the bind resolver(3C) library to gather count and time statistics. Otherwise, it is a straight-forward effort of parsing arguments and setting appropriate parameters. dig uses resolver routines res_init(), res_mkquery(), and res_send(), and it accesses the _res structure. You can compile dig with the standard resolver library, but this will change the output format, make the print options meaningless, and not gather RTT and packet count statistics.

BUGS

When you are running dig in batch mode, if an error occurs in one of the resolver routines, dig exits. The preferred behavior is to suspend only that particular query and to continue with the query list. The fix involves modifying the resolver routines to return a status that indicates and error, rather than simply exiting.

The ping option simply makes a call to the shell. This should be replaced with internal ping code.

FILES

/etc/resolv.conf	Initial domain name and name server addresses
LOCALRES	Environment variable that specifies the file to use in place of /etc/resolv.conf
LOCALDEF	Environment variable that specifies the default environment file

SEE ALSO

named(8), ping(8)

host(1B), nslookup(1) in the *UNICOS User Commands Reference Manual*, Cray Research publication SR–2011

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

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

RFC 1035, Domain Names–Implementation and Specification, Mockapetris, November 1987

NAME

diskusg – Generates disk accounting data by user, account, or group ID

SYNOPSIS

/usr/lib/acct/diskusg [-A] [-D] [-G] [-U|-a] [-z] [-i *ignorelist*] [-d *debuglevel*] [[-E] | [-F *s1*:*s2*:...*sn*] | [-T *t1*:*t2*:...*tn* [-t *timetype*]]] [-H] [-h] [-u *outputfile*] [-v] [-S] [-w *string*] [-x] [-y] *filesystems*

/usr/lib/acct/diskusg [-A] [-D] [-G] [-U|-a] [-i *ignorelist*] -s [-d *debuglevel*] [[-E] | [-F *s1*:*s2*:...*sn*] | [-T *t1*:*t2*:...*tn* [-t *timetype*]]] [-H] [-h] [-u *outputfile*] [*files*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The diskusg command generates disk accounting data that can used by the UNIX System V (standard) accounting system or by Cray system accounting (CSA). Disk accounting data is obtained either by scanning the inode region of the logical disks or by merging diskusg output records.

diskusg output, which is written to standard output (stdout), is usually the input to acctdisk(8). The acctdisk command converts diskusg -D output to tacct format, and converts -A and -U (or -a) output to cacct format; acctdisk does not convert -G output nor the output of options -E, -F, and -T. Standard accounting uses tacct format, while CSA uses cacct format. The tacct and cacct records can be merged for reporting with other accounting records of the same format. diskusg normally is run in the dodisk(8) procedure.

diskusg accepts four types of options:

- Consolidation options
- Data input options
- Data output options
- Site-reserved options

In the descriptions of these options, account ID (*acid*), group ID (*gid*), and user ID (*uid*) are numeric values, and account name (*account*), group name (*group*), and login name (*login*) are alphanumeric strings.

Consolidation Options

The following diskusg options allow you to consolidate information in different ways:

-A Reports the following disk usage information per account, which can be converted by acctdisk(8) to cacct format:

acid account #blocks #mblocks

For each account, the following information is provided:

acid Account ID.

account Account name.

#blocks Total number of disk blocks allocated to this account.

#mblocks Total number of migrated disk blocks allocated to this account. (This value will always be 0 if the Data Migration Facility (DMF) is not enabled in this file system.)

`-D` (Default) Prints the following on the standard output, one line per user, which can be converted by `acctdisk`(8) to `tacct` format:

uid login #blocks

For each user, the following information is provided:

uid User ID.

login Login name.

#blocks Total number of disk blocks allocated to this user.

`-G` Reports the following `diskusg` information per group, which cannot be processed by `acctdisk`(8):

gid group #blocks #mblocks

For each group, the following information is provided:

gid Group ID.

group Group name.

#blocks Total number of disk blocks allocated to this group.

#mblocks Total number of migrated disk blocks allocated to this group. (This value will always be 0 if DMF is not enabled in this file system.)

`-U` or `-a` Reports the following disk usage information per user ID, which can be converted by `acctdisk`(8) to `cacct` format:

uid login acid account #blocks #mblocks

The `-U` designation for this option supercedes `-a` and is the preferred usage. The information provided about the user/account pair is as follows:

uid User ID.

login Login name.

acid Account ID.

account Account name.

#blocks Total number of disk blocks allocated to this user/account pair.

#mblocks Total number of migrated disk blocks allocated to this user/account pair. (This value will always be 0 if DMF is not enabled in this file system.)

`-z` Allows the files with an account ID of 0 to be accumulated separately from the primary account for the user. By default, files with an account ID of 0 are accumulated to the primary account of the user. If a user does not have a primary account ID, then the account ID of 0 is used and always reported.

Data Input Options

The following options allow you to alter the input to `diskusg`:

`-i` *ignorelist*

Ignores file systems by these names. The names are those written to the super block by the `labelit`(8) command. The file system names in *ignorelist* must be separated by a comma or a blank space. If the list contains blank spaces, the list must be enclosed by double quotes.

`-s` Specifies that the input data is already in `diskusg` output-record format. The `-S` and `-s` options are mutually exclusive. Input is either from *files* or the standard input. The `diskusg` command combines all lines for a single user into a single line. If the input files were created by using any of the `-A`, `-D`, `-G`, or `-U|-a` options, that option must also be specified. The input files made with the `-h` or `-H` option have the character # placed at the start of the header lines. The # character as the first character of a line will cause the line to be treated as a comment and the line will be ignored.

Data Output Options

The following options allow you to alter the output produced by `diskusg`: The `-E`, `-F`, and `-T` options are all mutually exclusive, and these report formats cannot be processed by `acctdisk`(8).

`-d` *debuglevel*

Turns on debugging. Debugging levels 0 through 5 are available. Level 0 is the default and generates no debug messages, level 5 is the highest level and generates all available debugging messages. The debugging messages are sent to standard error (`stderr`).

`-E` Reports disk usage in an extended format for the report options (`-A`, `-D`, `-G`, or `-U|-a`). The `-E`, `-F`, and `-T` options are all mutually exclusive. For each record ID (account, group, or user ID), the following additional information is provided:

Report Item	Description
`Total Files`	Total number of inodes.
`Directory Files`	Total number of directory inodes.
`Online Files`	Total number of file inodes with online date.
`Offline Files`	Total number of file inodes with offline data.
`Total Blocks`	Total number of data blocks used.
`Online Blocks`	Total number of data blocks residing online.
`Offline Blocks`	Total number of data blocks residing offline.

`ACL Blocks`	Total number of blocks used by access control lists (ACLs).
`PAL Blocks`	Total number of blocks used by privilege assignment lists (PALs).

`-F` *s1*`:`*s2*`:...`*sn*

Reports disk usage per file size range (*s1* and so on) for the report options (`-A`, `-D`, `-G`, or `-U|-a`). The `-E`, `-F`, and `-T` options are all mutually exclusive. The file size ranges are specified by the site using this option. See Example 3. You may specify up to ten file size ranges; the sizes must be given in data blocks and must be in ascending order. (See NOTES for more information about data block sizes.) For each record ID, the following additional information is provided:

Report Item	**Description**
`Total Blocks`	Total number of data blocks used.
s1 `Blocks`	Number of data blocks used in the range greater than or equal to 0 and less than or equal to *s1*.
s2 `Blocks`	Number of data blocks used in the range greater than *s1* and less than or equal to *s2*.
sn `Blocks`	Number of data blocks used in the range greater than *sn-1* and less than or equal to *sn*.
`Max Size Blocks`	Number of data blocks greater than *sn* used.

`-T` *t1*`:`*t2*`:...`*tn*

Reports disk usage per file age range (*t1* and so on) for the report options (`-A`, `-D`, `-G`, or `-U|-a`). The `-E`, `-F`, and `-T` options are all mutually exclusive. The file age ranges are specified by the site using this option. See Example 4. You can combine the `-t` option with `-T` to select a particular time field. You may specify up to ten ages; the ages must be given in hours and must be in ascending order. For each record ID, the following additional information is provided:

Report Item	**Description**
`Total Blocks`	Total number of data blocks used.
t1 `Blocks`	Number of data blocks used in the range greater than or equal to 0 and less than or equal to *t1*.
t2 `Blocks`	Number of data blocks used in the range greater than *t1* and less than or equal to *t2*.
tn `Blocks`	Number of data blocks used in the range greater than *tn-1* and less than or equal to *tn*.
`Max Time Blocks`	Number of data blocks greater than *tn* used.

`-t` *timetype*
Must be used with the `-T` option. Specifies for the `-T` option the time field from the inode to use in the data collection. Possible values for *timetype* are:

`access` Uses the last access time of the data. This value is the default.

`inode` Uses the update time of the inode.

`update` Uses the update time of the data.

`-H` Displays an extended header. The header includes the data report header from the `-h` option, a report description, machine information, a report date line, a report format line. It also shows totals resulting from inode searches, including totals for unknown (charged to no one) accounts, users, or groups, depending upon what options you specified, and an overall total line. All of these lines start with a # character, so that they will be ignored as comments if the report is used as input to `diskusg` with the `-s` option.

`-h` Displays a report header that describes the report columns.

`-u` *outputfile*
Writes terse records to *outputfile* of files that are charged to no one (unknown); they display what is known about the inodes searched. Records include the disk–partition name, the inode number (*inode_num*), the user ID number (*uid*), the inode partition (*ipart_num*), the inode region (*ireg_num*) and the inode offset (*ioff_num*).

`-v` Sets verbose mode. A list of files that are charged to no one is written to `stderr`. This list includes files that have invalid user ID/account ID combinations.

Site-reserved Options

The following options have been reserved for site-specific use:

`-S` Site-reserved option that allows you to create a site-specific report format. The `-S` and `-s` options are mutually exclusive.

`-w` *string* Site-reserved option that accepts a string as input. The meaning of this option and accepted string values are determined by the site.

`-x` Site-reserved option for a True or False value. The option is True if it is present. The meaning of this option is determined by the site.

`-y` Site-reserved option for a True or False value. The option is True if it is present. The meaning of this option is determined by the site.

NOTES

Sites can tailor the contents of the reports by modifying the `site.c` module and recompiling and re-installing `diskusg`. (See the Site-reserved Options subsection.) For more information on how to use the site-reserved reports in the `site.c` source file, see *UNICOS Resource Administration*, Cray Research publication SG–2302.

In reporting the number of data blocks, the program has no way of differentiating among varying block sizes. Various devices may have different block sizes. Reports are comparable only if the underlying device uses the same block size.

EXAMPLES

Example 1: The following example displays the disk usage of `/` and `/usr`:

```
diskusg -U /dev/dsk/usr /dev/dsk/root > /usr/adm/acct/day/dtmp
```

Example 2: The following example generates an extended format report of usage by user ID with column titles:

```
diskusg -UEh /dev/dsk/ptmp
```

Example 3: The following example generates a file size format report of usage by group ID with column titles and extended header:

```
diskusg -Gh -F 100:1000:10000 /dev/dsk/ptmp
```

```
# Group ID      Group Name      Total   0 <    100   <   1000   <  10000   <    Max
#                               Blocks       Blocks     Blocks     Blocks     Blocks
         0            root     275568        32180      64984      46472     131932
         2             bin      26336          892        868      24576          0
         3             sys        144          144          0          0          0
         9        operator        280           80        200          0          0
       168             dce         40           40          0          0          0
      1007         craysrc       3228         2532        696          0          0
      1013              os      21392         4580      14064       2748          0
      1015         network      33344        12228       9408      11708          0
      1090             mpp          8            8          0          0          0
     11121         tsttool      46020        43292       2728          0          0
     11824         testing         32           32          0          0          0
     11951           vsxg0          8            8          0          0          0
     12584         craydev         44           44          0          0          0
```

Example 4: The following example generates a file age format report of usage by account ID with ages of 7 days, 30 days, and 180 days with report headers:

```
diskusg -AH -T 168:720:4320 /dev/dsk/ptmp
```

```
#  Disk Usage report by diskusg - Version 1: (SN-1703 (cool.cray.com)) on
#     Mon May  2 11:03:48 1994
#
#  Scanning filesystems:
#
#  /dev/dsk/ptmp
#
#
#  Account ID Usage report by diskusg - Version 1: (SN-1703 (cool.cray.com)) on
#     Mon May  2 11:03:48 1994
#
#  File Age Report (-T) - Selected by Access time, with the sample parameters in Hours.
#
# Account ID  Account Name      Total   0 <   168 H  <   720 H  <  4320 H  <    Max
#                              Blocks       Blocks     Blocks     Blocks     Blocks
         0         A-00000         16           16          0          0          0
      8306           Admin        400          296         44         40         20
      8359         Country         84           60          0          4         20
      8366          Cust_f          4            4          0          0          0
      8367           Demos         48           36          0         12          0
      8373          SysAdm         16           16          0          0          0
      8374            Intl        304          260          0         44          0
      8383          Netdev     116184        97132      18908        112         32
      8388          Syssup      11608        11304        304          0          0
      8394         Userint      42640        24020      13976       1728       2916
      8395           Users      48712        48460         12        240          0
      8397           Xydev     187516       164408      22344        180        584
#
# -- Totals ---               407532       346012      55588       2360       3572
#
```

FILES

`/etc/udb`	User validation file containing user control limits; used for user ID to login name conversions.
`/etc/acid`	Account ID information file containing account names; used for account ID to name conversions.
`/etc/group`	Account ID information file containing group names; used for group ID for name conversions.

SEE ALSO

`acct`(8), `acctdisk`(8), `acctsh`(8), `dodisk`(8), `labelit`(8), `mount`(8)

UNICOS Resource Administration, Cray Research publication SG–2302

NAME

dmap – Maps physical and logical devices

SYNOPSIS

```
/etc/dmap [-o] [-p name]
/etc/dmap [-o] [-l name]
/etc/dmap [-o] [-l minor_number]
```

IMPLEMENTATION

Cray PVP systems

DESCRIPTION

The dmap command produces a display providing information about the configuration of the disk subsystem. The type of display is determined by which of the following options and arguments you use with dmap:

`-o` Displays cylinder numbers in octal format. (Default is decimal.)

`-p` *name* Displays slices of all logical devices that reside on the named physical device. The display uses the following format:

```
minor  slice  cyls/block  blocks  total  name
-----  -----  ----------  ------  -----  ----
```

The format headings represent the following:

`minor`	Minor device number of logical device
`slice`	Slice index of logical device
`cyls/block`	Cylinder range on the physical device (block range for SSD and BMR)
`blocks`	Number of physical disk blocks
`total`	Cumulative block count
`name`	Name of logical device if established

`-l` *name*
or
`-l` *minor_number*

Displays all slices for the logical device specified by either its *name* (that is, `/dev/dsk/a`) or its *minor_number*. The display uses the following format:

```
slice  cyls/block  blocks  total  Physical device
-----  ----------  ------  -----  ---------------
```

The format headings represent the following:

`slice`	Slice index of logical device

`cyls/block`	Cylinder range on physical device (block range for SSD and BMR)
`blocks`	Number of physical disk blocks
`total`	Cumulative block count
`Physical device`	Name of physical device

NOTES

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

Active Category	Action
`system`, `secadm`	Allowed to use this command.
`sysadm`	Allowed to use this command. Shell redirected output is subject to security label restrictions.

If the `PRIV_SU` configuration option is enabled, the super user is allowed to use this command.

SEE ALSO

General UNICOS System Administration, Cray Research publication SG–2301

NAME

dnsquery – Queries domain name servers by using resolver library calls

SYNOPSIS

dnsquery [-n *nameserver*] [-t *type*] [-c *class*] [-r *retry*] [-p *retry period*] [-d] [-s] [-v] *host*

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The dnsquery program provides an interface to nameservers by using Berkeley Internet Name Daemon (BIND) resolver library calls. The program supports queries to the name server with an operation code of QUERY. This program is intended to be a replacement or supplement to programs like nstest, nsquery, and nslookup. All arguments except for *host* and *nameserver* are case-sensitive.

The dnsquery program accepts the following options:

-n *nameserver* Names the name server to be used in the query. Name servers can appear as either Internet addresses of the form *w.x.y.z* or can appear as domain names. The default is specified in the /etc/resolv.conf file.

-t *type* Specifies the type of resource record of interest. *Types* can be one of the following values. The default is ANY.

A	Address
NS	Nameserver
CNAME	Canonical name
PTR	Domain name pointer
SOA	Start of authority
WKS	Well-known service
HINFO	Host information
MINFO	Mailbox information
MX	Mail exchange
RP	Responsible person
MG	Mail group member
AFSDB	DCE or AFS server
ANY	Wildcard

The *type* argument is case-insensitive.

`-c` *class* Specifies the class of resource records of interest. The *class* argument can be one of the following values. The default is `IN`.

`IN`	Internet
`HS`	Hesiod
`CHAOS`	Chaos
`ANY`	Wildcard

The *class* argument is case-insensitive.

`-r` *retry* Specifies the number of times to retry if the name server is not responding. The default is 4.

`-p` *retry period* Specifies the period to wait before timing out. The default is `RES_TIMEOUT`.

`-d` Turns on debugging. This sets the `RES_DEBUG` bit of the resolver's options field. The default is debugging turned off.

`-s` Uses a stream rather than a packet. This uses a TCP stream connection with the name server rather than a UDP datagram. This sets the `RES_USEVC` bit of the resolver's options field. The default is a UDP datagram.

`-v` Synonym for the `s` flag.

host Specifies the name of the host (or domain) of interest.

MESSAGES

If the resolver fails to answer the query and debugging has not been turned on, `dnsquery` will print a message like the following:

```
Query failed (rc = 1) : Unknown host
```

The value of the return code is supplied by `h_errno`.

BUGS

Queries of a class other than `IN` can have interesting results because ordinarily a name server has only a list of root nameservers for class `IN` resource records.

Query uses a call to `inet_addr()` to determine if the argument for the `-n` option is a valid Internet address. Unfortunately, `inet_addr()` seems to cause a segmentation fault with some (bad) addresses (for example, `1.2.3.4.5`).

FILES

`/etc/resolv.conf`	To get the default name server and search lists
`arpa/nameser.h`	List of usable RR types and classes
`resolv.h`	List of resolver flags

SEE ALSO

`named`(8)

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

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

NAME

dodisk – Performs disk accounting

SYNOPSIS

/usr/lib/acct/dodisk [-a] [-v] [*special_files*]

/usr/lib/acct/dodisk [-A] [-v] [*special_files*]

/usr/lib/acct/dodisk [-o [*mount_points*]]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

By default, the dodisk command performs disk accounting on the special files listed in the /etc/checklist file. It is typically run by using the cron(8) command. checklist should contain a list of special files, one file name per line.

When running Cray Research system accounting (CSA), either the -a or -A option must be specified with dodisk. Failure to do so can cause CSA to abort.

The dodisk command accepts the following options and operands:

-a Reports disk usage in the following format:

uid login account acid #blocks

The information provided is as follows:

uid Numerical user ID of the user

login Login name of the user

account Account name

acid Account ID

#blocks Total number of disk blocks allocated to this user

-A Reports disk usage in the following format:

acid account #blocks

The information provided is as follows:

acid Account ID

account Account name

#blocks Total number of disk blocks allocated to this user

`-o` Causes `dodisk` to invoke `acctdusg` rather than `diskusg` to account for disk usage by login directory. This option causes `dodisk` to perform more slowly.

mount_points
Specifies one or more file system names, rather than special devices, where disk accounting is performed. If *mount_points* are not specified, "/" is the default file system.

`-v` Sets verbose mode. `dodisk` prints verbose records on standard error of files that are charged to no one.

special_files
Specifies one or more special files where disk accounting is performed.

NOTES

If you want to convert the output produced by `dodisk -A` to `cacct` format, you must use `acctdisk`(8) with the `-A` option.

EXAMPLES

The following example is a possible entry for the `/usr/spool/cron/crontabs/root` file so that `cron`(8) automatically runs `dodisk`:

```
30  10  *  *  1-5  /usr/lib/acct/dodisk -a -v 2> /usr/adm/acct/nite/dsklog
```

SEE ALSO

`acct`(8), `acctdisk`(8), `acctsh`(8), `cron`(8), `csa`(8), `diskusg`(8)

UNICOS Resource Administration, Cray Research publication SG–2302

NAME

dump – Invokes an incremental file system dump

SYNOPSIS

/etc/dump [-A *altfile*] [-a] [-B *buffer count*] [-c] [-d *density*] [-D *device*] [-e] [-f *file*] [-g *devgrp*] [-l *label_type*] [-m *capacity*] [-M *m_fmt*] [-n] [-P *p_fmt*] [-R *r_fmt*] [-s *size*] [-t *dump_level*] [-T *t_fmt*] [-u] [-v *vsn_list*] [-W] [-w] [-Z *max_list*] [-z] *filesystem*

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The dump command copies to magnetic tape or disk all files changed after a certain date in a specified *filesystem*, which is a required operand. (The date is found in the /etc/dumpdates file.)

The dump command accepts the following options:

-A *altfile* Specifies the name of a file to contain a second copy of the output from the beginning of dump. The EXAMPLES section shows how this command is used. You can use this file as input to /etc/restore -v to obtain, without a tape mount, a list of the files contained in the full dump output.

-a Dumps files copied by the Data Migration Facility (DMF) as if they were already migrated offline, that is, inode information only, without data. Use this option to avoid an unnecessary duplication of data when the data migration facility is managing the *filesystem*. Files that are eligible for this special processing include both dual-state files and the premigration copies of files fully backed up by DMF.

-B *buffer count* Determines the number of 32,678-byte blocks that are output in a single listio(2) operation. Default is 1 for pipes, and 20 for other output files.

-c Causes dump to use cartridge tapes rather than round tapes. This is equivalent to -g CART.

-d *density* Obsolete and optional. Valid arguments are 1600 or 6250. The default interpretation is the density of the nine-track output device. Thus, if the -d 1600 option is present along with -g TAPE, the capacity of one output volume is set to 40 Mbytes. Other values are ignored.

-D *device* Obsolete and optional. The default interpretation is the name of a specifically requested output device.

-e Requests a message to stderr at the approximate location of every volume change. The following is a sample output line produced by executing dump with this option:

```
dump (/root to root_dmp):  volume 1 ends near i-node 227
```

The approximate location is an estimate determined by the `-m` parameter, or by default calculation. Using this option facilitates skipping tapes during a restore operation that needs only one file.

Running `restore` with the `-t` option provides an inode to file name mapping that can help determine on which tape the file's data is located, when correlated with the `-e` option. If the `-f` option is used with the `-e` option, you must specify `-m` *capacity* for the command to supply location information.

`-f` *file* Places the dump on *file*, instead of tape. If you specify *file* as `-`, `dump` writes to standard output.

`-g` *devgrp* Specifies an explicit assignment to the character string that is used as the device group to reserve for the output device. The default value is `TAPE`.

`-l` *label_type* The *label_type* specified on the `-l` option is used in the default for the `-M` option. The tape daemon supports the following *label-type* values:

`nl` Nonlabeled tapes

`sl` IBM standard labels

`al` ANSI labels (default)

You cannot use the `-l` option with the `-f` option (dump to a file).

`-m` *capacity* Explicitly states in megabytes the capacity of a single output volume.

`-n` Obsolete and optional. Causes `dump` to broadcast one of the following messages by using `/etc/wall -g oper`:

When the program is aborting: `dump program is falling to floor in pieces`

When the program completes successfully: `dump has completed`

`-s` *size* Obsolete and optional. The default value is 2400. The default interpretation is the number of feet in an output device when `-g TAPE` is set.

`-t` *dump_level* Causes an incremental dump. Valid dump levels are 0 through 9; the default value is 9. If you specify 0, the entire file system is dumped. A level number above 0, incremental backup, tells `dump` to copy all files new or modified since the last dump of a lower level.

`-u` If the dump completes successfully, writes the date and time of the beginning of the dump on the `/etc/dumpdates` file. Lines in the `/etc/dumpdates` file are limited to 4096 characters.

`-v` *vsn_list* Causes `dump` to use *vsn_list* as a list of volume serial numbers (VSNs) to be used for the dump. Each VSN is a set of 1 to 6 alphanumeric characters separated by colons (:). You cannot use the `-v` option with the `-f` option (dump to a file). If `-v` is not used, `dump` asks the operator to type in a VSN list.

Example:

```
dump -v root1:root2:root3 /dev/dsk/root_fs
```

`-W` Tells the operator the file systems that must be dumped. (This information is gathered from the `/etc/dumpdates` and `/etc/fstab` files.) This option causes `dump` to print out, for each file system in `/etc/dumpdates`, the most recent dump date and level, and to highlight those file systems that should be dumped. If this option is set, all other options are ignored, and `dump` exits without further processing.

`-w` Similar to the `-W` option, but it prints only the file systems that must be dumped.

`-z` This option is the same as specifying `-Z 4096`.

`-Z` *max_list* Skips all regular nonmigrated files larger than the specified size.

The following four options are the format specifiers:

`-M` *m_fmt* Builds a `tpmnt` request. The default value is as follows:

```
tpmnt -r in  -g +g -v +i -f +I -n -l +l +?D'-D' +D +?d'-d' +d -P +O
```

`-P` *p_fmt* An option that is used as the prompt if a VSN list is requested from the operator. The default value is `please enter +e vsn's`. See the EXAMPLES section for uses of this option.

`-R` *r_fmt* Reserves an output device. The default value is `rsv +g`.

`-T` *t_fmt* Appends information to the end of the `/etc/dumpdates` line. The default value is +40i.

The format specifiers make use of a string substitution capability, based loosely on the format string for the `date` command. The following substitutions are recognized:

`+D` From the `-D` option, optional, may be `NULL` pointer.

`+d` From the `-d` option, optional, may be `NULL` pointer.

`+e` The estimated number of volumes.

`+f` The base name of the file system dumped.

`+g` The string from the `-g` option. Typical values are `TAPE`, `CART`, and `SILO`.

`+i` From the `-v` option or the operator input string.

`+I` Short form of `+i`, terminated at first colon (`:`).

`+l` From the `-l` option, typical values are `nl`, `sl`, and `al`.

`+O` The temporary file name for output (generated by the `dump` command).

`+t` The dump type. A single character from the `-t` option.

A number between the + character and the field identifier specifies a maximum length. The following conditional substitutions also are recognized:

`+?D_`*AA*`_` If the `-D` option has a value, the character string delimited by the _ characters is copied.

`+?d_`*AA*`_` If the `-d` option has a value, the character string delimited by the _ characters is copied.

`+?+_+_` The conditional `+?+` always tests true. This implements an escape mechanism. The example places a `+` in the output.

Because the tapes are under the control of the tape daemon, the `dump` program does not request operator assistance to handle tape errors. Disk errors cause a message to be printed and the buffer to be emptied. Disk errors do not cause the program to stop.

The tape interface logic of the `dump` command includes a `setjob`(2) system call. This permits independent `/bin/rsv` invocations and helps avoid reaching job tape limits when running multiple copies of the dump program from the same login.

When a `dump` is performed on an active file system, all files that satisfy the following two criteria will be available for restoration from the back-up medium:

- The file must be complete at the time of the `sync`() call at the beginning of the dump program.
- Neither the file, nor any components in the file path, may change until dump has completed.

NOTES

To use other, specific tape mount options not covered by `dump`, you can perform your own `rsv`(1) and `tpmnt`(1) operations, as follows:

```
rsv
tpmnt -l al -v a1:a2 -M -p /tmp/tapedev
dump -t 9 -u -f /tmp/tapedev /dev/dsk/slash_a
rls -a
```

Alternately, you can use the format specifiers. The following `dump` command alone yields the same results as the preceding example:

```
dump -t 9 -u -v a1:a2 -R "rsv" -M "tpmnt -l al -v +i -M -p +O" /dev/dsk/slash_a
```

On a file system with active data migration, the migrated files are not backed up, only moved. The `dump` command does not recall migrated data, but it does make a copy of the on-disk inode that contains the migration recall key. The `-a`, `-Z`, and `-z` options permit customer-specified processing on a system that has data migration. The capability also exists to uniquely specify desired policies in the `duex.c` user-exit source file.

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

Active Category	**Action**
`system`, `secadm`	Dump any file system.
`sysadm`	Dump any file system, subject to security label restrictions on the file system path and device, and the device labeling policy. Shell redirected output is subject to security label restrictions.

If the PRIV_SU configuration option is enabled, the super user is allowed to dump any file system.

CAUTIONS

The dump and restore(8) commands share an internal inode number, which is used to pass information between levels of an incremental restore. If an inode region is added to your file system, the internal number of a given inode may change, making an incremental restore impossible. If the super-block time-stamps indicate that this may have happened, the dump command prints a warning. If the maximum inode number has changed, an incremental restore fails; you can use restore -x or restore -i.

The restore program contains an error-recovery algorithm that searches forward in units of 4096 bytes for the beginning of a cu_spcl structure (declared in /usr/include/dumprestor.h). If the dump output is buffered or rebuffered using a buffer size that is not a multiple of 4096 bytes, and if one such buffer-full of data is lost, the error recovery algorithm of restore will be disabled.

EXAMPLES

Example 1: To do a full dump to cartridge tape of each file system weekly, with an incremental dump daily, perform the following:

Once a week (everything in mass storage), enter the following command:

```
dump -t 0 -u -c /dev/dsk/slash_a
```

Repeat for each file system.

Daily (everything in file system that has been modified):

```
dump -t 9 -u -c /dev/dsk/slash_a
```

Repeat for each file system.

Example 2: To write dumps to round tapes with IBM standard labels, using a specified volume serial number list (this example is an incremental dump):

```
dump -d 6250 -t 9 -l sl -v a1:a2 -u /dev/dsk/slash_a
```

Example 3: The following example shows the use of the -P option:

```
#  /etc/dump " -P "enter +e VSN(s) from VSN pool for filesystem +f" -u -t 9 /dev/dsk/dumptest
dump (/dumptest to tape): Date of this level 9 dump: Mon Mar  9 11:15:01 1992
dump (/dumptest to tape): Date of last level 3 dump: Mon Jan 27 15:35:11 1992
dump (/dumptest to tape): Dumping /dumptest
dump (/dumptest to tape): to tape
dump (/dumptest to tape): mapping (Pass I) [regular files]
dump (/dumptest to tape): mapping (Pass II) [directories]
dump (/dumptest to tape): estimated 3158 sectors on 0.08 volume(s).
dump (/dumptest to tape): enter 1 VSN(s) from VSN pool for filesystem /dumptest
```

At this point, the operator would enter a VSN list and the process would continue.

Example 4: The following example dumps the in-use inodes from `/dev/dsk/xx` into disk file `ofile` and lists the files on `stdout`:

```
/etc/mknod apipe p
/etc/restore -T -t -f apipe &
/etc/dump -t 0 -A apipe -f ofile /dev/dsk/xx
```

The `dump` and `restore`(8) programs work locally. The `rdump`(8) and `rrestore`(8) programs work when the tape drive can be accessed with `/etc/rmt`. When the Cray Research tape daemon owns the tape drive, the following procedure writes a dump from a network-connected machine:

```
here> rsv CART
here> tpmnt -n -P tname ...
here> remsh there /etc/dump -t0 -f- /dev/dsk/xx  "|"  dd bs=409600 >tname
here> rls -a
```

FILES

`/etc/dumpdates`	Dump date record
`/etc/fstab`	File system information (used by the `-w` and `-W` options)

SEE ALSO

`rdump`(8), `restore`(8), `tpdaemon`(8)

`rls`(1), `rsv`(1), `tpmnt`(1) in the *UNICOS User Commands Reference Manual*, Cray Research publication SR–2011

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

General UNICOS System Administration, Cray Research publication SG–2301

NAME

ebmchk – Scans an a.out file looking for EBM instructions

SYNOPSIS

/etc/ebmchk [-v] [-o *a.out_output_file*] *a.out_input_file*

IMPLEMENTATION

CRAY C90 series

DESCRIPTION

The ebmchk command scans the code space in an a.out file looking for enable bidirectional memory (EBM) instructions. If an EBM instruction is found in a valid code sequence, ebmchk prints the parcel address (relative to the beginning of the code segment); otherwise, it silently terminates.

The ebmchk command accepts the following options:

-v Prints all occurrences of the EBM pattern and a description of how ebmchk interpreted it. For actual EBM instructions, it also finds and prints the name of the preceding entry point from the traceback records, in addition to the parcel address specified by the default.

-o *a.out_output_file*
Creates an a.out file that is identical to the original a.out file with EBM instructions replaced by DBM instructions.

EXAMPLES

Example 1: To execute ebmchk on all files in /bin, enter the following:

```
$ su
Password:
# find /bin -type f -exec ./ebmchk {} ;
```

Example 2: To check an a.out file for occurrences of the EBM instruction (in verbose mode), enter the following:

```
ebmchk -v a.out
```

Example 3: To replace occurrences of the EBM instruction with DBM instructions, enter the following:

```
ebmchk -o b.out a.out
```

NAME

econfig – Verifies boot-time CSL directives for IOS model E systems and generates mknod commands for specified configuration

SYNOPSIS

/etc/econfig [-d] [-n] [-v] [*pfile*]

IMPLEMENTATION

Cray PVP systems with an I/O subsystem model E

CRAY J90 series

CRAY EL series

DESCRIPTION

The econfig command performs bounds checking (gaps and overlaps) and some validity and syntax checks on boot-time configuration specification language (CSL) directives read from its standard input or from the file *pfile*. This command is used to generate mknod requests that reflect the device configuration defined in the *pfile* (deadstart parameter file).

The econfig command accepts the following options and operand:

-d Generates mknod commands on stdout necessary to create the file system described in *pfile*.

-n Do not perform validity checking on device definitions in the supplied parameter file.

-v Generates comment headers on stdout describing the type of mknod commands that follow.

pfile File (that is, a deadstart parameter file) that contains CSL directives that describe a file system.

NOTES

If econfig returns a large number of error messages, they are most likely due to a "cascade effect" where the first few lines are valid errors, and the rest are "errors" due to those first few lines not being correct. For example, if the configuration of an I/O cluster is incorrect, this is listed as an error, but so is every subsequent reference to that cluster; simply correcting the cluster's configuration will cause the other reference errors to disappear. If you get a long error listing, concentrate on correcting the first few errors listed; then try it again.

CAUTIONS

The econfig utility does not check for overlapping disk slices.

NAME

errdemon – Invokes the error-logging daemon

SYNOPSIS

/etc/errdemon [*file*]

IMPLEMENTATION

Cray PVP systems

DESCRIPTION

The error-logging daemon errdemon collects error records from the operating system by reading special file /dev/error and places them in *file*. No analysis of the error records is done by errdemon; that responsibility is left to errpt(8) and/or the olhpa(8) diagnostic utility.

errdemon will launch the monitoring and notification program watchstream(8) if the file /etc/watcherror.re exists.

The daemon is terminated by errstop(8). Only an appropriately authorized user can start the daemon, and only one daemon can be active at any time.

file Specifies an output file for errdemon. If *file* does not exist, it will be created; otherwise, error records are appended to it, so that no previous error data is lost. Default: /usr/adm/errfile.

NOTES

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

Active Category	Action
system, secadm, sysadm	Allowed to start the error-logging daemon.

If the PRIV_SU configuration option is enabled, the super user is allowed to start the error-logging daemon.

FILES

/dev/error	Source of error records
/usr/adm/errfile	Default repository for error records

SEE ALSO

`errpt`(8) for information on processing the error report generated by `errdemon`
`errstop`(8) for information on terminating the error-logging daemon
`olhpa`(8) for information on the system error-log formatter
`thresholding`(7) for an introduction to automated monitoring and notification
`watchstream`(8) for information on the monitoring and notification program

`err`(4) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014, for information on the error-logging interface

UNICOS Resource Administration, Cray Research publication SG–2302
Tape Subsystem Administration, Cray Research publication SG–2307

System Maintenance and Remote Testing Environment (SMARTE) Guide, Cray Research publication SD–1017. (This document contains information private to Cray Research, Inc. It can be distributed to non-CRI personnel only with approval of the appropriate Cray manager.)

NAME

errpt – Processes errors report generated by errdemon(8)

SYNOPSIS

/etc/errpt [-s *date*] [-s l*number*] [-e *date*] [-e l*number*] [-a] [-d *pdevlist*] [-f] [-h] [-o] [-q] [-r] [-t] [*files*]

IMPLEMENTATION

Cray PVP systems

DESCRIPTION

The errpt command processes data collected by the error-logging mechanism (errdemon(8)) and generates a report of that data. The default report is a detailed listing of errors posted in the *files* specified on the command line. Options apply to all files. If you do not specify *files*, errpt will try to use /usr/adm/errfile as *files*.

A summary report listing the options that might limit its completeness and the time stamped on the earliest and latest errors encountered is appended to the error report.

Memory location on the CRAY J90 series is reported as three values (for example, 01/01/05) corresponding to CRAY J90s with a 2X2, 4X4, and 8X8 backplane (respectively). The correct memory location value should be selected based on the hardware configuration of the machine.

A report can be limited to certain records by the invocation of any of the following options:

-s *date* Ignores all records posted earlier than *date*. *date* has the format *mmddhhmmyy*[*ss*] and is consistent in meaning with the date(1) command. The first *mm* is the month number; the second *mm* is the minute number.

-s l*number* Specifies that an error report start l*number* days prior to the date and time specified by either of the -e options. If the -e *date* or -e l*number* option is not specified, the start time and date are set to *number* days less than the current time and date. Note that this argument consists of a lowercase letter "l" (not the number "1"), with no space between the "l" and the following *number* value.

-e *date* Ignores all records posted later than *date*. *date* has the format *mmddhhmmyy*[*ss*] and is consistent in meaning with the date(1) command. The first *mm* is the month number; the second *mm* is the minute number.

-e l*number* Specifies that an error report end l*number* days prior to the current date and time. Note that this argument consists of a lowercase letter "l" (not the number "1"), with no space between the "l" and the following *number* value.

-a Produces a detailed report that includes all error types.

`-d` *pdevlist* Limits a detailed report to data about devices given in *pdevlist*. *pdevlist* can be one of two forms: a list of device identifiers separated from one another by commas, or a list of device identifiers enclosed in double quotation marks and also separated from one another by commas. `errpt` is familiar with the following device types: `comm`, `fddi`, `hippi`, `tape`, `lsp`, `dsk`, `ethernet`, and `atm`. (`dsk` includes only driver error records and not detailed IOS error records.) (See the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014.) For a list of supported disk devices, see `diskspec`(7). The device type `ios` implies all devices connected to the IOS (all of the preceding identifiers). An additional identifier is `mem`, which includes a detailed report of memory errors.

For IOS model E systems, the `-d` *pdevlist* option does not display the relative block number of a file system. The `bb`(8) command must be used to find the relative block number.

`-f` In a detailed report, limits the reporting of device errors to unrecovered errors.

`-h` In a detailed report, limits the reporting of errors to actual hardware device errors.

`-o` Produces a one line summary for each error. The summaries are sorted by device type and also sorted chronologically within each device type.

`-q` Produces a Quick Summary Report of all errors in the specified files. The Quick Summary includes an error total for each type of device; that is, memory, disk, and tape. For some devices, the total unrecovered errors are also displayed.

`-r` Formats the error records in raw mode. This will be useful if the error log has errors from devices or machine types that `errpt` is not prepared to format. You can also use the `-r` option to ensure that `errpt` is formatting new records properly.

`-t` Requests the formatted output of the buffered logs from the control units of the IBM ESCON 3490, 3490E, and 3590 devices in addition to the other output requested. There is no way to print only the buffered log records.

files Specifies the files that `errpt` will process. If *files* is not specified, `errpt` tries to use `/usr/adm/errfile`.

NOTES

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

Active Category	Action
`system`, `secadm`, `sysadm`	Allowed to use this command.

If the `PRIV_SU` configuration option is enabled, the super user is allowed to use this command.

Two types of error packets are associated with disk errors: C packets and A packets. C packets are sent by the IOS to log all disk errors. A packets are sent by the IOS in response to an I/O completion. The `errpt` program reports A-packet errors when no options are used; `errpt` reports both A- and C-packet errors when the `-a` option is used.

Recovered read errors will be reported in the A-packet response. However, because all write operations are write behind, recovered write errors are not reported in the A-packet response.

FILES

`/usr/adm/errfile`	Default error file
`/usr/src/uts/sys/erec.h`	Error record include file

SEE ALSO

`bb`(8) for information on creating relative bad block files from ASCII flaw table files
`errdemon`(8) for information on invoking the error-logging daemon

`date`(1) for information on printing and setting the date in the *UNICOS User Commands Reference Manual*, Cray Research publication SR–2011

`errfile`(5) for information on error log file format in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

`diskspec`(7) for information on the physical characteristics of disk drives on Cray PVP systems (available only online)

General UNICOS System Administration, Cray Research publication SG–2301
UNICOS Resource Administration, Cray Research publication SG–2302
Tape Subsystem Administration, Cray Research publication SG–2307

NAME

errstop – Terminates the error-logging daemon

SYNOPSIS

/etc/errstop

IMPLEMENTATION

Cray PVP systems

DESCRIPTION

The errstop command sends a software termination signal (SIGTERM) to the error-logging daemon errdemon(8). Only a super user can terminate the error-logging daemon. errstop also terminates the diagnostic daemon dgdemon(8).

SEE ALSO

dgdemon(8) for information on invoking the diagnostic daemon
errdemon(8) for information on invoking the error-logging daemon

NAME

esdmon – Interactively monitors the logical-layer External Semaphore Device

SYNOPSIS

/etc/esdmon [-c] [-d] [-h] [-l] [-m] [-n *unicos-name*] [-p] [-r *repeat-interval*] [-s *SFS-Arbiter*]

IMPLEMENTATION

Cray PVP systems

DESCRIPTION

The esdmon command is used to interactively monitor the logical-layer External Semaphore Device. esdmon uses the special devices /dev/sfs and /dev/smp to interface to the respective device drivers.

Executing the esdmon command requires super-user permissions.

-c (Continuous) Causes esdmon to use the *curses* library for full-screen displays. Usually, esdmon operates in line mode. When invoked in *continuous* display mode, the initial display shows the *heart-beat* and *port* tables, and a two-line synopsis of the External Semaphore Device assignment. At the end of each *repeat-interval*, keyboard input is sampled for one of the following commands:

- ^l — Clear and refresh the display.
- \+ — Increase the *repeat-interval* by 1 second.
- \- — Decrease the *repeat-interval* by 1 second.
- > — Move the the next higher display.
- < — Move the the next lower display.
- 0-4 — Move directly to the indicated display.
  - 0 — The initial display shows the *heart-beat* and *port* tables, and a two-line synopsis of the External Semaphore Device assignment.
  - 1 — Display the first of three lock assignments displays.
  - 2 — Display the second of three lock assignments displays.
  - 3 — Display the third of three lock assignments displays.
  - 4 — A full screen display of all available state information for all 64 semaphores in the External Semaphore Device.
- h,? — Display the *help* menu.
- q — Quit, exit the program.

-d (Debug) Causes extra information to be displayed. Usually, this information is not needed and can interfere with other displays.

`-h` Displays the *heart-beat* table.

`-l` Displays the *lock* table.

`-m` Displays the *media* table.

`-n` *unicos-name*
Allows for an alternate UNICOS binary to be used for the *nlist* symbol table search.

`-p` Displays the *port* table.

`-r` *repeat-interval*
When used with the `-c` option, specifies the *repeat* interval used for refreshing the full-screen updates. The default value is 2 seconds.

`-s` *SFS-Arbiter*
Specifies the name of the *SFS Arbitration Service* to be monitored.

The *SFS-Arbiter* name must match one of the valid entries in the `/etc/config/sfs` configuration file.

FILES

`/etc/config/sfs` The Shared File System configuration file.

SEE ALSO

`sfs`(4) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

NAME

exportfs – Exports and unexports directories to NFS clients

SYNOPSIS

/etc/exportfs [-a] [-i] [-o *options*] [-u] [-v] [*pathname*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The exportfs command makes a local directory or file available for mounting over the network by network file system (NFS) clients. The exportfs command cannot export remote mount file systems. It usually is invoked at boot time by an entry in the /etc/config/daemons file, and it uses information that is contained in the /etc/exports file to export *pathname* (which you must specify as a full path name). Only appropriately authorized users can run exportfs at any time to alter the list or characteristics of exported directories and files. Directories and files that are currently exported are listed in the /etc/xtab file.

With no options or arguments, exportfs prints the list of directories and files that are currently exported.

The exportfs command accepts the following options:

-a Exports all *pathnames* that are listed in /etc/exports, or if you specify -u, unexports all of the currently exported *pathnames*.

-i Ignores the options in /etc/exports. Usually, exportfs consults /etc/exports for the options that are associated with the exported *pathname*.

-o *options* Specifies a comma-separated list of optional characteristics for the *pathname* that is being exported. You can select the *options* from among the following:

access=*client*[:*client*]...
Gives mount access to each client that is listed. The default value allows any machine to mount the specified directory.

anon=*uid* Specifies *uid* as the effective user ID when a request comes from an unknown user. Root users (*uid* 0) are always considered unknown by the NFS server, unless they are included in the root option (following). The default value for this option is –2. To disable anonymous access, and to allow clients to mount only the exported directory, set the value of anon to –1.

cksum Checksums packets that are returned to clients.

kerberos
Specifies that Kerberos Remote Procedure Call (RPC) (AUTH_KERB) messages should be used for NFS transactions. The kerberos option may not be used with the "krb" option.

`krb` Specifies Kerberos authentication that is required for access to this export.

`nosync` Specifies that write operations to this file system are delayed. This option can significantly improve write performance but its use can result in loss of data if the server crashes before the data is written to disk.

`ro` Exports *pathname* with read-only characteristics. If you omit this option, *pathname* is exported with read-write characteristics.

`root=`*hostname*[:*hostname*]...
Gives root access only to the root users from a specified *hostname*. The default is that no hosts are granted root access.

`rw=`*hostname*[:*hostname*]...
Exports *pathname* with read-mostly characteristics. *Read-mostly* means exported with read-only characteristics to most machines, but with read-write characteristics to those specified. If you omit this option, *pathname* is exported with read-write characteristics to all.

`-u` Unexports the indicated *pathnames*.

`-v` Prints each directory or file as it is exported or unexported (verbose).

pathname Specifies path name.

NOTES

You cannot export a directory that is either a parent directory or a subdirectory of one that is currently exported and within the same file system.

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

Active Category	Action
`system`, `secadm`	Allowed to use this command.
`sysadm`	Allowed to use this command. Shell redirected I/O is subject to security label restrictions.

If the `PRIV_SU` configuration option is enabled, the super user is allowed to use this command.

The UNICOS kernel must be configured with `NFS_SECURE_EXPORT_OK`.

The *client* argument can specify the name of a host or the name of a netgroup. `netgroups` is not supported for the `hostname` field in `/etc/exportfs`. For information on how to use a netgroup file, see `netgroup`(5).

FILES

`/etc/exports`	Static export information
`/etc/xtab`	Current state of exported *pathnames*

SEE ALSO

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

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

UNICOS Networking Facilities Administrator's Guide, Cray Research publication SG–2304

NAME

`fddidump` – Utility to dump FDDI (FCA-1) shared memory

SYNOPSIS

`/etc/fddidump` [`-a` *address*] [`-l` *length*]

IMPLEMENTATION

Cray PVP systems with I/O subsystem model E

DESCRIPTION

The `fddidump` utility dumps the contents of shared memory on an FDDI (FCA-1) channel adapter. To perform this function, `fddidump` uses the ioctl interface to the FDDI driver (`fd.c`) running on the Cray Research system.

Command-line options are as follows:

`-a` *address* Sets the shared memory address from which the dump will begin. Default is 0x1000.

`-l` *length* Sets the length (in bytes) of the dump. Default is 512.

SEE ALSO

CRI documents for FDDI:

`xfddidump`(8)

`fddi`(4) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

ANSI documents for FDDI:

FDDI MAC (Media Access Protocol) Specification (FDDI-MAC), document number X3.139–1987, November 5, 1986

FDDI PHY (Physical Layer Protocol) Specification (FDDI-PHY), document number X3.148–1988, June 30, 1988

FDDI PMD (Physical Medium Dependent) Specification (FDDI-PMD), document number X3.166–1990, September 28, 1989

FDDI SMT (Station Management) Specification (FDDI-SMT), document number X3T9.5/84–49, Rev 6.2, May 18, 1990

Other related documents for FDDI:

RFC 1188 Proposed standard for the transmission of IP datagrams over FDDI networks. October 1990; D. Katz

Logical Link Control Specification (802.2 LLC), document number 802.2–1985, July 16, 1984

NAME

`fddiload` – Utility to load FDDI (FCA-1) microcode

SYNOPSIS

`/etc/fddiload` [`-b` *binary*] [`-z` *device*]

IMPLEMENTATION

Cray PVP systems with I/O subsystem model E

DESCRIPTION

The `fddiload` utility loads microcode running on the FDDI (FCA-1) channel adapter into its shared memory. To perform this function, `fddiload` uses the ioctl interface to the FDDI driver (`fd.c`) that runs on the Cray Research system.

The `fddiload` utility accepts the following options:

`-b` *binary* Sets the full path name to the file containing the binary image of the FCA-1 microcode. Default is `/etc//micro/FCA1.ucode`.

`-z` *device* Sets the UNICOS device name for the driver. Default is `/dev/fddi0/fd00`.

NOTES

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

Active Category	Action
`system`, `secadm`, `sysadm`	Allowed to use this command.

If the `PRIV_SU` configuration option is enabled, the super user is allowed to use this command.

SEE ALSO

CRI documents for FDDI:

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

`fddi`(4) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

ANSI documents for FDDI:

FDDI MAC (Media Access Protocol) Specification (FDDI-MAC), document number X3.139–1987, November 5, 1986

FDDI PHY (Physical Layer Protocol) Specification (FDDI-PHY), document number X3.148–1988, June 30, 1988

FDDI PMD (Physical Medium Dependent) Specification (FDDI-PMD), document number X3.166–1990, September 28, 1989

FDDI SMT (Station Management) Specification (FDDI-SMT), document number X3T9.5/84–49, Rev 6.2, May 18, 1990

Other related documents for FDDI:

RFC 1188 Proposed standard for the transmission of IP datagrams over FDDI networks. October 1990; D. Katz

Logical Link Control Specification (802.2 LLC), document number 802.2–1985, July 16, 1984

NAME

`fddimap` – Utility to gather FDDI station management information

SYNOPSIS

`/etc/fddimap` [`-d`] [`-e`] [`-p`] [`-u` *port*] [`-i` *infc*] [`-r`] [`-c`] [`-S`] [`-N`] [`-m` *meth*] [`-s` *name* | `MAC address`]

IMPLEMENTATION

Cray PVP systems with I/O subsystem model E

DESCRIPTION

The `fddimap` utility transmits FDDI station management (SMT) frames onto an FDDI ring from a Cray Research system that is directly attached to that same FDDI ring, to gather information about a particular station or all stations on the FDDI network. This utility also can build a logical ring map of the FDDI network.

To perform its function, `fddimap` uses the utility socket interface to the station management daemon (SMTD) running on the same Cray Research system (see `smtd`(8)).

The `fddimap` utility accepts the following options:

`-d` Enables debug mode; verbose output.

`-e` Sends a single SMT ECHO packet to the specified station.

`-p` Sends a continuous set of SMT ECHO packets to the specified station. This is very similar to a `/etc/ping` command, except it does not involve the IP protocol.

`-u` *port* Sets the UDP port number to which socket requests are sent. Default is 3000. See `smtd`(8).

`-i` *infc* Sets the interface ordinal of the FDDI interface to use when sending requests on the FDDI ring. Default is 0.

`-r` Prints a logical ring dump of all of the stations that exist on the FDDI ring.

`-c` Retrieves the MAC counters from the specified station. See option `-s`.

`-S` Retrieves the configuration SIF and operation SIF information from the specified station. See option `-s`.

`-N` Retrieves the NIF information from the specified station. See option `-s`.

`-m` *meth* Prints a logical ring map of the FDDI ring by using one of two methods specified by `meth`. Method 1 sends NIF requests counter-clockwise around the ring, starting with this station's upstream neighbor. Method 2 sends SIF requests clockwise around the ring, starting with this station's downstream station.

`-s` *name* | `MAC address`
Specifies the name (or FDDI MAC address) to which to send requests. (See options `-c`, `-S`, `-N`, `-e`, and `-p`.) When specifying a MAC address, use hexadecimal addresses in Canonical form, with colon (:) separators. (See the MAC Address File subsection.)

The MAC Address File

Rather than having to remember the 48-bit MAC addresses of all FDDI stations on the ring, you can create a file that contains these addresses (similar to the `/etc/hosts` file) to give names to each of the FDDI stations on the ring. This file is called `/etc/ethers`, and is standard on most UNIX based systems. The format of the file is as follows:

```
#
#    The addresses in this file must be in Canonical (Ethernet) form.
#
0:40:a6:0:0:e0          sn1703b-1
00:40:a6:00:00:10       sn1703b-2
ff:ff:ff:ff:ff:ff       broadcast
```

Comment lines may be placed anywhere in the file as long as the comments starts in the first column of a line. Comments are designated by a # character in column 1. Leading zeros within each byte are not required, but do make it easier to read for the users.

FILES

`/etc/ethers` File that contains MAC addresses of all FDDI stations on an FDDI ring.

SEE ALSO

`smtd`(8), `xfddimap`(8)

`fddi`(4) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

The ANSI documents for FDDI:

FDDI MAC (Media Access Protocol) Specification (FDDI-MAC), document number X3.139–1987, November 5, 1986

FDDI PHY (Physical Layer Protocol) Specification (FDDI-PHY), document number X3.148–1988, June 30, 1988

FDDI PMD (Physical Medium Dependent) Specification (FDDI-PMD), document number X3.166–1990, September 28, 1989

FDDI SMT (Station Management) Specification (FDDI-SMT), document number X3T9.5/84–49, Rev 6.2, May 18, 1990

Other related documents for FDDI:

RFC 1188 Proposed standard for the transmission of IP datagrams over FDDI networks. October 1990; D. Katz

Logical Link Control Specification (802.2 LLC), document number 802.2–1985, July 16, 1984

NAME

`fddistat` – Displays information about IOS model E FDDI devices

SYNOPSIS

`/etc/fddistat`

IMPLEMENTATION

Cray PVP systems with IOS model E

DESCRIPTION

The `fddistat` utility gets information from the UNICOS FDDI driver tables and from the Connection Management (CMT) processor on the FCA-1 FDDI channel adapter in an IOS model E. Information is obtained by using an `ioctl` system call and is displayed on the user's screen. For proper viewing, you need a display of at least 40 lines and 80 columns.

Command abbreviations are shown on the bottom line of each `fddistat` screen. All `fddistat` commands are a single character in length and are case-sensitive. Commands are as follows:

`h` Displays the help menu.

`r` Refreshes the display.

`e` Exits the `fddistat` program.

`s` Displays device and logical path statistics.

`p` Displays boot-time and run-time device and logical path parameters.

`q` Displays logical path read and write buffer queues.

`d` Sets device ordinal or displays device configuration and general state information.

`l` Sets logical path ordinal or displays logical path information.

`f` Displays information about the actual FDDI interface.

`x` Toggles extended view on and off.

`D` Enables device ordinal selection by using the plus (`+`) or minus (`-`) key.

`L` Enables logical path selection by using the plus (`+`) or minus (`-`) key.

`+` Increments device or logical path ordinal.

`-` Decrements device or logical path ordinal.

`>` Increases screen refresh rate.

`<` Decreases screen refresh rate.

Selecting Screen Refresh Rate and Ordinals

Near the bottom of the screen, just above the command help line, `fddistat` displays the current screen refresh rate, the current device ordinal, and the current logical path ordinal. You can change these settings at any time.

To select a new screen refresh rate, simply press the less than (`<`) or greater than (`>`) key to decrease or increase the rate.

Notice that either Selected Device or Selected Logical Path is highlighted; the one that is highlighted is the one that will be changed when you press the `+` or `-` key. Enter `D` on the command line to select the device ordinal or enter `L` to select logical path. After selecting one of these choices, use the `+` or `-` key to increment or decrement the ordinal.

Extended View

For debugging purposes, some fields are displayed in hexadecimal form when in extended mode.

`Statistics` Screen

Statistics are shown for both the device as a whole (left) and for the selected logical path (right). The `Statistics` screen displays the following items:

Item	Description
`Read Msgs`	Number of messages (frames) received from the FDDI network. Instantaneous read message rate is shown in parenthesis.
`Read Bytes`	Number of bytes received from the FDDI network. Instantaneous read bytes are shown in parenthesis.
`Write Msgs`	Number of messages (frames) sent to the FDDI network. Instantaneous write message rate is shown in parenthesis.
`Write Bytes`	Number of bytes sent to the FDDI network. Instantaneous write bytes are shown in parenthesis.
`Read Timeouts`	Number of times the IOS has returned a read request with a read time-out error code.
`Write Timeouts`	Number of times the IOS has returned a write request with a write time-out error code.

`No. Read Errors`
`No. Write Errors`
`No. Misc Errors`
	Total number of read, write, and miscellaneous errors that have occurred.

`User Read Errors`
`User Write Errors`
`User Misc Errors`
	The last four read, write, and miscellaneous error codes returned to the user in `errno`.

`Spec Read Errors` `Spec Write Errors`	

`Spec Misc Errors`	The last four read, write, and miscellaneous specific error codes that occurred. These errors go together with the user error codes. The specific error code is a more descriptive error code than `errno`.

`Parameters` Screen

Two versions of parameters are shown, `RUN` and `BOOT`. The `RUN` version shows the value of the parameter as it exists in the running system. The `BOOT` version shows the value of the parameter when the system was booted. If they are different, an `ioctl` was used by some process to change them. The `Parameters` screen displays the following items:

Item	Description
`Device TREQ`	Requested value for the target token-rotation timer (TTRT) that this station issues in its claim frames.
`Device Copy Criteria`	Value written to the copy criteria register in the channel adapter. (See the `fddi`(4) man page for a further explanation.)
`Device Pad Count`	Number of bytes the channel adapter hardware inserts at the beginning of all frames received and stripped from the beginning of all frames transmitted.
`Device Max Write`	Maximum number of write requests that can be posted to the IOS before queuing occurs in the mainframe driver.
`Device Max Reads`	Maximum number of reads requests that can be posted to the IOS before queuing occurs in the mainframe driver.
`Path RFT`	List of all frame types that the logical path receives.
`Path Options`	List of currently active options on the logical path.
`Path Read Timeout`	Value for read time-outs used on the logical path.

`Queues` Screen

The `Queues` screen shows a summary of the read and write buffer queues for the logical path. A buffer is either a read or write request. The `Queues` screen displays the following items:

Item	Description
`No. Active`	Number of buffers (I/O requests) currently active in the IOS.
`No. Queued`	Number of buffers (I/O requests) currently queued in the mainframe and waiting to be issued to the IOS.
`First`	Address of the first buffer (I/O request) on queue.
`Last`	Address of the last buffer (I/O request) on queue.
`Next`	Address of the next buffer (I/O request) on queue. This is the next buffer that will be issued to the IOS on completion of a buffer of the same type.

Device Screen

If you enter the Device command followed by a number, the ordinal of the selected device is set to that number (for example, d 1 sets the device ordinal to 1). The Device screen shows the following items:

Item	Description
IOPATH	I/O cluster, IOP, or IOP channel in which the FDDI interface is installed.
Flags	Currently active device flags. The FDDI driver uses these flags internally to coordinate its activities.
MAC Address *(IEEE)*	48-bit Media Access Control (MAC) address of an interface in Canonical form (the form in which the address appears in Cray main memory).
MAC Address *(FDDI)*	Media Access Control (MAC) address of an interface in FDDI (MSB) form (the form in which the address appears when on the FDDI medium).
UNA Address *(IEEE)*	48-bit Media Access Control (MAC) address of an interface's Upstream Neighbor in IEEE form.
UNA Address *(FDDI)*	48-bit Media Access Control (MAC) address of an interface's Upstream Neighbor in FDDI form.
Old UNA Address *(IEEE)*	48-bit Media Access Control (MAC) address of an interface's previous Upstream Neighbor in IEEE form.
Old UNA Address *(FDDI)*	48-bit Media Access Control (MAC) address of an interface's previous Upstream Neighbor in FDDI form.
DNA Address *(IEEE)*	48-bit Media Access Control (MAC) address of an interface's Downstream Neighbor in IEEE form.
DNA Address *(FDDI)*	48-bit Media Access Control (MAC) address of an interface's Downstream Neighbor in FDDI form.
Old DNA Address *(IEEE)*	48-bit Media Access Control (MAC) address of an interface's previous Downstream Neighbor in IEEE form.
Old DNA Address *(FDDI)*	48-bit Media Access Control (MAC) address of an interface's previous Downstream Neighbor in FDDI form.
MAC Services Available	Indicates whether MAC services are available; if not available, the MAC is offline for some reason. When offline, frames cannot be sent or received.
LLC Services Available	Indicates whether LLC services are available; if LLC services are not available, protocols such as TCP/IP cannot use the network interface.

`DAD Results` (MAC_DA_Flag)
: Shows results of the FDDI Duplicate Address Detection test, which is constantly being executed as a part of Station Management (SMT).

`No. Logical Paths Configured`
: Shows maximum number of logical paths configured on the device. This number usually comes from the parameter file at boot time.

`No. Logical Paths Open`
: Shows current number of logical paths open on the device.

`Paths That Are Open`
: Shows current list of logical paths open on the device.

`Logical Path` Screen

If you enter the `Logical Path` command followed by a number, the ordinal of the selected path is set to that number (for example, `l 5` sets the path ordinal to `5`). The `Logical Path` screen displays the following items:

Item	Description
`Flags`	Logical path flags that currently are active. The FDDI driver uses these flags internally to coordinate its activities.
`PID`	Process ID of the process that currently has the logical path open.
`Command`	UNICOS command name that currently has the logical path open. This should correspond to the PID.

`Fddi` Screen

The `Fddi` screen displays information about the state of the actual FDDI interface on the channel adapter. A brief summary of what is displayed in this screen follows. For a more detailed description of these parameters, see the ANSI FDDI SMT (Station Management) Specification.

Item	Description
`T_MAX`	Maximum target token-rotation timer (TTRT) that this station supports.
`TVX`	Maximum time in which this station should see a valid frame or token.
`T_REQ`	Value transmitted in this station's claim frames. This is the token-rotation timer (TRT) at which this station desires the ring to operate.
`T_NEG`	Value that the claim process negotiates. This is a result of all stations claiming with their respective `T_REQ` or a higher claim.
`CFM State`	Current state of the configuration management (CFM) state machine for the station. CFM manages the internal token path within the station.
`RMT State`	Current state of the ring management (RMT) state machine for the station. RMT manages the state of the ring at all times and determines when the ring is operational. RMT also is responsible for recovering the ring once it has seen a catastrophic fault.

`UNA` MAC address of this station's upstream neighbor shown in both IEEE and FDDI forms.

`MY Address` MAC address of this station shown in both IEEE and FDDI forms.

`DNA` MAC address of this station's downstream neighbor shown in both IEEE and FDDI forms.

`TX_Ct` (Frames Transmitted Count)
Number of frames that this station transmitted successfully.

`Token_Ct` (Token Count)
Number of valid tokens that this station received.

`RX_Ct` (Frames Received Count)
Count of all complete frames received, including MAC frames, void frames, and frames that this station stripped.

`Late_Ct` (Late Count)
When the ring is operational, number of times TRT expired before the ring became operational the last time. When the ring is nonoperational, number of times TRT has expired while waiting for the ring to become operational. This count remains 15 until a MAC reset occurs.

`Copied_Ct` (Frames Copied Count)
Number of frames this station copied successfully.

`Lost_Ct` (Lost Frame Count)
Number of instances in which a format error is detected in a frame or token jeopardizing credibility of PDU reception.

`NotCopied_Ct` (Frames Not Copied Count)
Number of frames intended for this station but not copied successfully.

`Error_Ct` (Error Isolated Count)
Number of error frames this station detected that no previous station detected. Error frames may be any of the following types: FCS error detected and received E indicator clear; frame of a length that is not valid and received E indicator clear; or received E indicator not equal to `SET` or `RESET`.

`Ring_Ct` Number of times the ring has gone from not operational to operational.

`TVX_Ct` Number of times TVX has expired. TVX is the FDDI timer that ensures a valid frame or token is received periodically.

`CEM State` Current state of the configuration element management (CEM) machine for the given port. CEM performs the interconnection of PHYs and MACs to configure the ports and MACs within the station.

`ECM State` Current state of the entity coordination management (ECM) state machine for the given port. ECM controls the optical bypass switch of the physical medium dependent (PMD) layer and signals the physical connection management (PCM) when the media is available. ECM also starts the PCM's for the A and B ports when the optical bypass switching is complete.

`Current RX LS`
Sampling of the FDDI line state currently being received by the given PHY.

`PCM State` Current state of the physical connection management (PCM) machine for the given port. PCM initializes the connection of neighboring ports and manages signaling between ports.

`T_VAL` Hexadecimal value representing bits transmitted to the neighboring PHY during the `SIGNAL` state of the PCM state machine. This code describes different things about this station to the neighboring station (for example, the port type (A, B, M, or S) of this port, the type of link confidence test (LCT) that this port requires (short, long, medium, or extended, and so on).

`R_VAL` Hexadecimal value representing bits received from the neighboring PHY during the `SIGNAL` state of the PCM machine.

`Mode` Type of attachment (`TREE` or `PEER`) to the ring. `PEER` mode indicates the station is connected to another port of similar type. `TREE` mode indicates the station is connected to a concentrator's M port; and thus, is part of a branch of a `TREE` off the dual ring.

`Neighbor Type`
Port type (A, B, M, or S) of the neighboring PHY.

`LEM Count` Number of link errors detected on this port since the station was last reset.

`LEM Reject Count`
Number of times this port was removed from the ring due to exceeding the LER threshold.

`LER Avg.` Long-term average link error rate estimate for this port. This count ranges from 10 e–4 through 10 e –15 bits.

`LER Alarm` Link error rate at which this link connection exceeds a preset alarm threshold. The alarm value ranges from 10 e–4 through 10 e –15 bits.

`LER Cutoff` Link error rate at which this link connection is flagged as faulty. The cut-off value ranges from 10 e–4 through 10 e –15 bits.

`EBError Count` (Elasticity Buffer Error Count)
Number of times the elasticity buffer overflowed.

SEE ALSO

`fddimap`(8), `xfddimap`(8), `smtd`(8)

`fddi`(4) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

The ANSI documents for FDDI:

FDDI MAC (Media Access Protocol) Specification (FDDI-MAC), document number X3.139–1987, November 5, 1986

FDDI PHY (Physical Layer Protocol) Specification (FDDI-PHY), document number X3.148–1988, June 30, 1988

FDDI PMD (Physical Medium Dependent) Specification (FDDI-PMD), document number X3.166–1990, September 28, 1989

FDDI SMT (Station Management) Specification (FDDI-SMT), document number X3T9.5/84–49, Rev 6.2, May 18, 1990

Other documents related to FDDI:

RFC 1188 Proposed standard for the transmission of IP datagrams over FDDI networks. October 1990. D. Katz

Logical Link Control Specification (802.2 LLC), document number 802.2–1985, July 16, 1984

NAME

ff – Lists file names and statistics for a file system

SYNOPSIS

/etc/ff [-d] [-a *n*] [-c *n*] [-i *inode_list*] [-I] [-l] [-m *n*] [-n *file*] [-p *prefix*] [-s] [-u] *special*

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The ff command (fast find) reads the i-list and directories of the special file *special*, assuming it is a file system, saving inode data for files that match the selection criteria. The *special* file can be a block special file (see mknod(8)). Output consists of the path name for each saved inode, plus any other file information requested, using the options below. Output fields are positional. Except for hard links, which are printed at the end, the output is produced in inode order; fields are separated by tabs. The number of files selected and, optionally, the number of hard links detected are sent as error output on separate lines. The default line produced by ff is as follows:

path-name i-number

With all options enabled, output fields area are as follows:

path-name i-number size uid

The *n* argument in the following option descriptions is used as a decimal integer. Where *n* indicates number of days, a day is defined as a 24-hour period and the *n* argument may be signed, with the following meanings when *n*, for example, has a value of 7:

+7 Equal to or more than seven days

7 Between seven and eight days

–7 Less than seven days.

The ff command accepts the following options:

-d Enables additional debugging messages.

-a *n* Selects if the inode has been accessed in *n* days.

-c *n* Selects if the inode has been changed in *n* days.

-i *inode_list* Generates names for only those inodes specified in *inode_list*.

-I Does not print the inode number after each path name.

-l Generates a supplementary list of all path names for multiply linked files.

`-m` *n*	Selects if the inode has been modified in *n* days.
`-n` *file*	Selects if the inode has been modified more recently than the *file* argument.
`-p` *prefix*	Adds the specified *prefix* to each generated path name. The default is "`.`".
`-s`	Prints the file size, in bytes, after each path name.
`-u`	Prints the owner's login name after each path name.
special	Name of the file system that contains the i-list and directories to be read by the `ff` command.

NOTES

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

Active Category	Action
`system`, `secadm`	Allowed to specify any file system.
`sysadm`	Allowed to specify any file system. Shell redirected output is subject to security label restrictions.

If the `PRIV_SU` configuration option is enabled, the super user is allowed to specify any file system.

BUGS

Only a single path name out of any possible path names is generated for a multiply linked inode, unless you specify the `-l` option. When you specify `-l`, no selection criteria apply to the names generated. All possible names for every linked file on the file system is included in the output.

EXAMPLES

Example 1: In the following example, `ff` generates a list of the names of all files on the `/dev/dsk/root` file system:

```
ff -I -l /dev/dsk/root
```

Example 2: In the following example, `ff` produces an index of files and inode numbers that are on the `/dev/dsk/usr` file system and have been modified in the last 24 hours:

```
ff -m -1 -l /dev/dsk/usr
```

Example 3: In the following example, `ff` obtains the path names for inodes 451 and 76 on the `/dev/dsk/usr` file system:

```
ff -i 451,76 /dev/dsk/usr
```

SEE ALSO

`mknod`(8)

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

NAME

fingerd – Daemon program for finger(1B)

SYNOPSIS

/etc/fingerd

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The fingerd command is the daemon program for the remote finger(1B) utility. It responds to requests for active user information by using the Internet finger protocol.

The inetd(8) command invokes fingerd. inetd listens on multiple ports and when a request is made on the finger port, it forks fingerd. fingerd uses the TCP port number 79 by default, as specified in the services(5) database.

FILES

/etc/inetd.conf	inetd configuration file that contains the fingerd entry
/etc/services	File that maps service names to port numbers

SEE ALSO

inetd(8)

finger(1B) in the *UNICOS User Commands Reference Manual*, Cray Research publication SR–2011

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

NAME

`fping` – Send an echo packet to a GigaRing node

SYNOPSIS

`/etc/fping` [`-c` *count*] [`-i` *wait*] [`-q`] [`-s` *size*] *special*

IMPLEMENTATION

CRAY T90 systems with GigaRing-based I/O

CRAY J90 systems with GigaRing-based I/O

DESCRIPTION

The `fping` command sends a series of echo message packets to the GigaRing node represented by the specified *special* file. These special files are, by convention, located in the `/dev/fmsg` directory; see `fmsg`(4) for further information. Upon receiving the echo packet, the remote GigaRing node will echo the packet back to the sender. The `fping` command prints out timing information for each echo packet sent.

The `fping` command accepts the following arguments:

`-c` *count* Specifies how many echo packets to send. The `fping` command stops after upon reaching this count. The default count is 8.

`-i` *wait* Wait *interval* number of seconds between each packet. The default interval is 1 second.

`-q` Quiet output. Just the summary line is displayed upon termination.

`-s` *size* Specifies the size of the payload, in 64 bit words, of the message packets sent. The default payload size is 1 word but the echo packet protocol and time stamps use an additional 4 words. The maximum value for size is 120.

special File representing the GigaRing node.

EXAMPLES

The following command sends 8 echo packets to GigaRing I/O node (ION) at ring 2, node 2:

```
# /etc/fping /dev/fmsg/e0202
4+1 words from 020: seq = 0: time = 164.082 + 173.033 us
4+1 words from 020: seq = 1: time = 174.838 + 173.620 us
4+1 words from 020: seq = 2: time = 173.576 + 176.560 us
4+1 words from 020: seq = 3: time = 174.527 + 152.162 us
4+1 words from 020: seq = 4: time = 171.096 + 173.013 us
4+1 words from 020: seq = 5: time = 174.411 + 153.058 us
4+1 words from 020: seq = 6: time = 172.447 + 174.411 us
4+1 words from 020: seq = 7: time = 173.042 + 152.998 us
FPING /dev/fmsg/e0202 (20): 5 data words
```

The `fping` command prints out 2 timing components. The first measures the time it takes to send the echo message packet from the `fmsg` driver in the Unicos kernel and receive the response back at the `fmsg` driver. This time includes Unicos message overhead, GigaRing message time to send the message packet, I/O node overhead to receive and retransmit the message, GigaRing message time for the message packet return, system interuupt overhead, and Unicos message overhead to receive the message. The second time component is the user system call time (a write) plus the time to reconnect the user (the `fping` command). Both times are in microseconds.

FILES

`/dev/fmsg/*`

SEE ALSO

`fmsg`(4) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

NAME

fsck – Checks file system consistency and interactively repairs it

SYNOPSIS

/etc/fsck [-c] [-d] [-f] [-i] [-n] [-p] [-q] [-s] [-S] [-u] [-W] [-y] *special*

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The fsck command audits and interactively repairs inconsistent conditions in an NC1FS file system.

If the file system is not consistent, fsck asks for confirmation before attempting any correction. In all cases, a y or yes response causes the correction to be made; a response of n or no leaves the file system unchanged. If you do not have write permission, fsck may ask for approval to change the file system, but the action is refused by the I/O subroutine after it tests a software lock.

Before a file system is mounted for general read/write usage, all of the questions asked by fsck should be answered affirmatively. An answer of n or no is appropriate for surveying a file system that will be corrected later. For example, you might answer questions with n or no if you wish to mount the file system as read-only to rescue some files before fsck causes them to be deleted or if you wish to repair the file system manually.

At the conclusion of the program, fsck summarizes the size of, and the free space in, the file system.

Some forms of disk corruption require that you run the fsck command several times.

The fsck command accepts the following options and operand:

-c Used in conjunction with -y, assumes a clear response to all yes, no, or clear questions that fsck asks.

-d Debug. Causes fsck to obtain the device characteristics from a prompted dialogue on stdin, rather than from a stat(2) system call. This permits fsck to be tested against something less than a mountable file system.

-f Performs a fast check. Checks only the inodes and allocations. Duplicate allocations are announced, but no inodes may be cleared without examination of the directories. The free lists of the dynamic block may be rebuilt.

-i Causes fsck to ignore most duplicate allocations. This option allows you to isolate corrupt inodes that cause multiple duplicate allocations. If you specify this option, you should run fsck again after clearing the corrupt inodes.

-n Assumes a no response to all questions that fsck asks.

-p Specifies "preen" mode, which is designed to match the mfsck(8) program interface.

`-q` Suppresses ("quiets") most of the `fsck` output, including warnings and errors that apply to specific files and directories, all errors from phase 5, and all problems with I/O. If you specify this option, and if no responses from the operator are required, the program writes less than twenty lines of output.

`-s` Ignores the actual free lists and (unconditionally) constructs new ones by rewriting the dynamic blocks of the devices.

`-S` Conditionally reconstructs the free list. Specifically, if no problems were detected in the file system, the free list is rebuilt. The test is made against a flag set whenever `fsck` asks a question.

`-u` Does not quit after reading the dynamic block, even if the file system appears to have been unmounted cleanly.

`-W` Suppresses all inode warning messages. You can generate and correct the warning-level problems after the file system is mounted.

`-y` Assumes a `yes` response to all questions that `fsck` asks.

special Name of the special file or the file system descriptor file that may be used to open the file system. A required operand.

The `fsck` command performs the following functions.

1. Verifies that *special* is a file system:
   - Locates super blocks and dynamic blocks
   - Verifies that inode blocks, bad blocks, super blocks, and dynamic blocks may be allocated without error
   - Verifies that total inode count is consistent with blocks allocated for inodes
   - Verifies that file system sizes are consistent with the value returned by the `stat`(2) system call
2. Examines each active inode:
   - Verifies that the mode field is valid
   - Verifies that file sectors may be allocated without conflict and within valid areas of the file system
   - Verifies that the last byte of the file is contained within the last allocation
3. Examines directory inodes:
   - Verifies pointers and signatures (the `cd_signature` field) in each directory sector
   - Verifies for each entry in a directory that a nonzero inode field refers to an accessible inode
   - Validates the directory tree structure
   - Checks directory allocation for missing sectors
4. Scans for directory errors. Unlinked directories are offered for inclusion in the `lost+found` directory. Other directory problems may require that the directory be cleared.
5. Performs a final pass through the active inodes:

- Offers unlinked nondirectory files for inclusion in the `lost+found` directory. Other file problems may require that the file be cleared.
- Verifies link count field of inodes.

6. Verifies dynamic information.
7. Rebuilds all dynamic information.
8. Terminates; prints a summary of the file system state.

NOTES

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

Active Category	Action
`system`, `secadm`, `sysadm`, `sysops`	Allowed to use this command.

If the `PRIV_SU` configuration option is enabled, the super user is allowed to use this command.

CAUTIONS

Some inconsistencies are noted by `fsck` but are not corrected. The `last byte not in last allocation` condition may be caused by pre-allocating a scratch file, and can be cleared by copying the file. The `. . .directory tree structure. . .` error may be caused by a root-user using `/etc/link` and `/etc/unlink`. It can be cleared in the same fashion. Neither of these conditions need prevent the file system from being mounted.

Most corrective actions result in some loss of data; you can determine the amount of data lost and the severity of the loss from the diagnostic output.

Any file system other than the root may be examined by `fsck` while it is mounted, but it will not be changed unless it is unmounted. The root file system is the only one `fsck` will change while mounted. On Cray PVP systems, the reboot step will be requested if necessary.

If unlinked inodes are relinked, existing space in an existing `lost+found` directory in the root directory is required.

FILES

`/dev/dsk/*` Block special device names

SEE ALSO

`crash`(8), `mdd_pre`(8), `mfsck`(8), `mkfs`(8), `mknod`(8)

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

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

General UNICOS System Administration, Cray Research publication SG–2301

NAME

fsdaemon – File system monitor daemon

SYNOPSIS

/etc/fsdaemon [-c *crit_cmd*] [-l] [-n [+]*incr*] [-p *path*] [-q] [-s *seconds*] [-w *warn_cmd*]

/etc/fsdaemon [-c *crit_cmd*] [-n [+]*incr*] [-p *path*] [-q] [-s *seconds*] [-t] [-w *warn_cmd*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The fsdaemon daemon monitors file systems and starts warning or critical commands if the amount of space in use exceeds a threshold value. When the daemon is initiated, it has no information about which file systems to monitor or what the warning or critical thresholds might be. The fsmon(8) command must be used to configure the daemon as needed. See the fsmon(8) man page in conjunction with this man page because these two programs are components of a single feature.

The fsdaemon daemon accepts the following options and arguments:

-c *crit_cmd* Causes the daemon to execute a command, *crit_cmd*, when a critical threshold is reached on a file system. Each occurrence of a critical condition results in the invocation of a separate instance of *crit_cmd* with the name of the file system (*filesystem*) appended to the *crit_cmd* string preceded by one space. After a command is started, the daemon no longer monitors that file system until a reset is received. To resume monitoring, the command must at some point execute the fsmon command with the -r or -R option and the file system name.

-l Locks daemon in memory. If this option is specified, the daemon locks itself in memory after it has finished initialization. This option is illegal if the -t option is used.

-n [+]*incr* Executes a nice(2) system call with the value -*incr* unless *incr* is preceded by +. For example, an *incr* value of 4 executes a nice(-4) system call and an *incr* value of +4 executes a nice(4) system call.

-p *path* Uses the specified *path*, rather than the default path, for the pipes and files used in communication between fsdaemon and fsmon. This allows you to test a file system monitor version without affecting a production version that may be running.

-q Allows the operator to terminate fsdaemon by using fsmon with the -q option. If this option is not specified, the daemon cannot be terminated with fsmon.

-s *seconds* Specifies cycle time between file system checks. The default is 5 seconds.

`-t` Test mode. Allows the daemon program to be run in the background by a user during development and testing, without the program executing system calls that can be used only by super users. If this option is specified, `fsdaemon` does not detach itself from the initiating terminal or become a true daemon.

`-w` *warn_cmd* Causes the daemon to execute *warn_cmd* when a warning threshold is reached on a file system. Each occurrence of a warning condition results in the invocation of a separate instance of *warn_cmd* with the name of the file system (*filesystem*) appended to the *warn_cmd* string preceded by one space. After a command is started, the daemon no longer monitors that file system until a `reset` is received. To resume monitoring, the command must at some point execute the `fsmon` command with the `-r` or `-R` option and the file system name.

NOTES

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

Active Category	Action
`system`, `secadm`, `sysadm`	Allowed to start the message daemon.

If the `PRIV_SU` configuration option is enabled, the super user is allowed to start the message daemon.

MESSAGES

The usual usage messages are written to the `stderr` file during the initialization phase. After daemon state is reached, all messages are written either to the log file or to the command output pipe.

FILES

`/usr/spool/fsmonitor/Fd.log`	Active log file
`/usr/spool/fsmonitor/Fsdmn.cmd_to`	Daemon command input pipe
`/usr/spool/fsmonitor/Fsdaemon.pid`	Daemon's process ID
`/usr/tmp/Fm_in`*xxxxx*	Daemon command output pipe

SEE ALSO

`fsmon`(8)

UNICOS Resource Administration, Cray Research publication SG–2302

NAME

fsed – Debugs NC1FS file systems

SYNOPSIS

/etc/fsed [*file1 file2* ...]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The fsed command debugs NC1FS file systems. It provides display and patch capabilities inside the files specified on the command line.

The input parse mechanism separates commands by using new lines and semicolons. Within a command, blanks and tab characters are used to separate tokens. A # character begins a comment. Most commands consist of a 2-character command code, followed by an optional modifier character and a single token command operand.

The program maintains six special addresses. The dot-address (.) is where the next display will begin. The double-quotation address (") is the result of the last address expression typed. All changes and name references use the double-quotation address. The d-address, the i-address, the SB-address, and the DB-address refer to the last locations that were displayed in the directory, inode, super-block, and dynamic-block format, respectively. These addresses are displayed in response to the id command with an a modifier.

The program maintains a single sector-sized buffer. The data in this buffer may be changed and examined repeatedly without being written to the disk. If the data in the buffer was changed, issuing a command that will destroy this data provokes an expected write message.

Commands

The fsed command contains the following commands:

he — Help. A command description is sent to standard output.

gd*mm nn* — General display. The next *nn* items at dot-address are displayed in a format specified by the modifier *mm*. The gd command accepts multiple and contradictory modifier characters. The default modifier is ow. Other modifier characters are as follows:

- o,O — Display in octal.
- x,X — Display in hexadecimal.
- d,D — Display in decimal.
- c,C — Display as characters, made printable.
- f,F — Display as floating point, 8 bytes per item.

b,B Each item consists of 1 byte.

p,P,q,Q Each item consists of 2 bytes.

h,H Each item consists of 4 bytes.

w,W Each item consists of 8 bytes.

fd*m nn* Formatted display. The next *nn* items at dot-address are displayed in a special format as indicated by the modifier *m* . The modifier can have any of the following values:

i,I Display as an inode.

d,D Display as a directory.

SB Display as a super block. Modifiers SBa, SBb, and SBc cause different parts of this information to be displayed.

DB Display as a dynamic block. The four dynamic block displays are DBa, DBb, DBc, and DBd.

(NULL) Repeats the last command if it was a display.

da*m expr* Defines address. If there is no modifier, *m*, the expression is a byte address within the current partition. The modifier can have the following values:

b The expression is a block number.

i The expression is an inode number.

n The expression is a name of a field in the structure at "- address.

Expressions may begin with a special address designator: ., ", i, or d.

gc *expr* General change. If *expr* is a string, the strlen(*expr*) bytes at double-quotation address are changed. If *expr* is not a string, a single word at double-quotation address (") is changed.

nc *name expr* Named change. The field named *name* at double-quotation address is changed to the new value, *expr*. Special-case coding provides for changing the signature in a directory entry with nc signature; no *expr* is required.

wr Write. The sector buffer containing possibly changed data is returned to the disk. You must use a write command to make these changes permanent.

nm*m req* Names. nm shows the list of recognized names. The optional modifier, *m*, (which can be any optional modifier) results in a full listing. The optional request, *req*, may have the following values:

i Inode names

d Directory names

SB Super-block names

DB Dynamic block names

`ni` *pathname*	Name inode. Begin in the root directory. Set dot address, double-quotation address, and inode address to the inode for the specified file by using the *pathname*.
`cp` *dev*	Change partition. Each file is identified by the device number in its super block. On CRAY Y-MP systems, you can use the `cp` command to examine several file systems simultaneously.
`id`*m*	Internal displays. The required modifier, *m*, can have the following values: `a` Designates the internal addresses `i` Designates the inode information for the current partition `p` Designates the partition summary
`fo` *path*	File out. The file from the current inode address is copied to the file that *path* specifies.
`op`*m path*	Open. An external file is made available by using a `fopen`(*path*, *m*) call. The `cl`, `ou`, and `in` commands also reference this external file.
`cl`	Close. The current external file is closed.
`ou`	Out. The sector buffer contents are written to the external file.
`in`	In. The sector buffer is filled with data from the external file.
`!` *arguments*	Escape to shell. The indicated *arguments* are sent to the system subroutine to be performed by a separate shell.
`qu`	Quit. The program performs the requested action.

MESSAGES

When an error occurs, an informative message is printed and the input buffer is flushed.

EXAMPLES

Example 1: The following example reads the `mbox` file in directory `/xxx` on Cray PVP systems:

```
ni /xxx/mbox;dan a0.blk;gdc 200
```

Repeated new lines will display successive characters.

Example 2: The following example restores the super block in sector 1 from a copy in sector 8, using the file `mike`:

```
opw mike; dab 8; ou ; cl
opr mike; dab 1; in ; wr; cl
```

Example 3: The following example restores the magic word in a dynamic block on Cray PVP systems:

```
dab DB; nc magic 0x6e6331646231636e
wr
```

NAME

`fslogd` – File system error logging daemon

SYNOPSIS

`/etc/fslogd`

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The file system error logging daemon, `fslogd`, collects file system error records from the operating system by reading the character special pseudo device `/dev/fslog`. It prompts the operator for desired recovery actions depending on the type of error encountered. These recovery actions allow increased system resiliency by enabling the system to continue running after file system errors have been detected. This mode of operation is selectable on a file system basis when you use the `mkfs`(8) or `setfs`(8) commands to disable the file system panic flag.

`fslogd` handles three types of errors: file system errors, directory errors, and inode errors detected by the kernel. File system errors cause `fslogd` to prompt the operator to request that the system be panic'ed, the corrupted file system data structure contents be formatted and displayed, or the file system in error be unmounted for maintenance by `fsck`(8) or `fsed`(8). The operator should choose the appropriate option based on the critical level of the file system involved and the extent of the damage. There are currently no directory errors reported by the kernel in `/dev/fslog`. The `fslogd` daemon reports inode errors to the operator to request that the system be panic'ed, the corrupted inode data structure contents be formatted and displayed, or the associated file in error be removed.

NOTES

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

Active Category	**Action**
`system`, `secadm`, `sysadm`	Allowed to use this command.

If the `PRIV_SU` configuration option is enabled, the super user is allowed to use this command.

FILES

`/dev/fslog`	Source of file system error log records

SEE ALSO

`mkfs`(8), `setfs`(8)

`fslog`(4), `fslrec`(5), `inittab`(5) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

General UNICOS System Administration, Cray Research publication SG–2301

NAME

`fsmap` – Displays file system free-blocks

SYNOPSIS

`/etc/fsmap` *fsname*

IMPLEMENTATION

Cray PVP systems

DESCRIPTION

The `fsmap` routine provides a display of the available free-blocks on a specified file system, *fsname*.

The format `fsmap` produces is as follows:

```
start  block   start  block   start  block   start  block
-----  ----- | -----  ----- | -----  ----- | -----  -----
```

Entries are displayed horizontally in four columns to conserve space. Each entry consists of a starting block number (`start`) and the number of contiguous blocks (`block`).

NOTES

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

Active Category	Action
`system`, `secadm`	Allowed to specify any file system.
`sysadm`	Allowed to specify any file system. Shell redirected output is subject to security label restrictions.

If the `PRIV_SU` configuration option is enabled, the super user is allowed to specify any file system.

EXAMPLES

Example 1: The following example shows a common use of `fsmap`:

```
fsmap /dev/dsk/usr_mail

Free block layout for: /dev/dsk/usr_mail

  Start    Count     Start    Count     Start    Count     Start    Count
 -------  ------- | -------  ------- | -------  ------- | -------  -------

Partition: 0  Blks: 0-4559  Dev: 40-A1-27

      5        1 |      38        1 |     102        1 |     105        1
    110        1 |     144        1 |     166        1 |     178        1
    237        3 |     243        1 |     245        3 |     250        1
    253     1239 |    1513      903 |    2417        1 |    2419      966
   3406       64 |    3472      361 |    3854      623 |    4478       82

Partition: 1  Blks: 4560-18239  Dev: 40-A1-27

   4873        3 |    4884        1 |    4886        9 |    4896        2
   4910     4717 |    9648        1 |    9651        6 |    9658        5
   9670        3 |    9678        2 |    9682        1 |    9684        9
   9812       12 |    9946        2 |   10124        2 |   10129        2
  10133        4 |   10142       46 |   10189       43 |   10253       10
  10426       42 |   10510      147 |   10678      183 |   10862     5863
  16726      133 |   16860     1380
```

Example 2: The next example shows another use of `fsmap`:

```
fsmap /dev/dsk/qtest3

Free block layout for: /dev/dsk/qtest3

 Start    Count     Start    Count     Start    Count     Start    Count
-------  ------- | -------  ------- | -------  ------- | -------  -------

Partition: 0  Blks: 0-10943  Dev: 40-A2-30

    186     6047 |    6234     2802 |    9037        5 |    9043      132
   9176     1768
```

When there is more than one physical partition in the specified logical device, then the message "`Partition: 0  Blks:#-# Dev:` *device_name*" is repeated for each partition.

SEE ALSO

`bmap`(8), `dmap`(8)

NAME

fsmon – Interfaces with the file system monitor fsdaemon(8)

SYNOPSIS

```
/etc/fsmon -a [-c nnn] [-d] [-D] [-i cw] [-p path] [-t] [-w nnn] filesystems
/etc/fsmon -a [-c nnn] [-d] [-D] [-n cw] [-p path] [-t] [-w nnn] filesystems
/etc/fsmon -a [-c nnn] [-D] [-e] [-i cw] [-p path] [-t] [-w nnn] filesystems
/etc/fsmon -a [-c nnn] [-D] [-e] [-n cw] [-p path] [-t] [-w nnn] filesystems
/etc/fsmon -m [-c nnn] [-d] [-D] [-i cw] [-f state] [-p path] [-t] [-w nnn] filesystems
/etc/fsmon -m [-c nnn] [-d] [-D] [-n cw] [-f state] [-p path] [-t] [-w nnn] filesystems
/etc/fsmon -m [-c nnn] [-D] [-e] [-i cw] [-f state] [-p path] [-t] [-w nnn] filesystems
/etc/fsmon -m [-c nnn] [-D] [-e] [-n cw] [-f state] [-p path] [-t] [-w nnn] filesystems

/etc/fsmon [-D] -q [-p path] [-t]
/etc/fsmon [-D] [-p path] [-t] -R filesystems
/etc/fsmon [-D] [-p path] [-t] -r state filesystems

/etc/fsmon [-D] -l [-p path] [-t]
/etc/fsmon [-D] [-p path] [-s state] [-t] [filesystems]
```

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The fsmon command is the command-level interface to the file system monitor daemon fsdaemon(8). It provides three functions:

- Adds or modifies information used by the daemon to monitor file systems. The first set of eight command lines in the SYNOPSIS section provides these functions.
- Affects the internal state of the daemon. The second set of three command lines in the SYNOPSIS section provides these functions.
- Displays log or file system status. Status displays can be selected based on state (using the -s option) or name. The third set of two command lines in the SYNOPSIS section provides these functions.

The fsmon command accepts the following options:

`-a` Adds specified *filesystems* to the list of monitored file systems. Each file system receives the critical threshold value (specified with the `-c` option), warning threshold value (specified with the `-w` option), and enabled state (specified with the `-e` option) or disabled state (specified with the `-d` option) supplied on the command line. (See those options for defaults and value ranges.) The special file system name `all` implies those file systems found in the kernel mount table, and only file systems not already in the table of monitored file systems are added. It is an error to add a file system that exists in the daemon's table. The `-a` option cannot be used with the `-m` option.

`-c` *nnn* Specifies the critical threshold value. This value specifies the percentage of allocated file system space, which, when reached, triggers the execution of the critical command. (The critical command is specified when `fsdaemon`(8) is initiated.) The value range is 0.0 < *nnn* < 100.0; the default value is 95.0%.

`-d` Disables monitoring of the specified *filesystems*. This option must be used with either the `-a` or `-m` options and cannot be used with the `-e` option. File system monitoring is enabled by default.

`-D` Specifies debug mode. This option may be used with any other option.

`-e` Enables monitoring of the specified *filesystems*. This option is the default; it must be used with either the `-a` or `-m` options and cannot be used with the `-d` option.

`-f` *state* Forces critical (`c`) or warning (`w`) state on the specified *filesystems*. This option is valid only with the `-m` option and causes the same action by the daemon as if the file system had actually reached the critical or warning threshold. If both the `c` and `w` values are supplied, both conditions are forced; however, the daemon acts only on the critical threshold because it checks for that condition first.

`-i` *cw* Specifies that an operator message is sent for the specified state. *cw* takes the following values:

- `c` Specifies that a message is sent for the critical state.
- `w` Specifies that a message is sent for the warning state.
- `cw` Specifies that a message is sent for both the critical and warning states. This is the default action if `-i` and `-n` options are not specified.

`-l` Displays the log file. The current content of the log file is written to the `stdout` file.

`-m` Modifies specified entries in the list of monitored file systems. One or more *filesystems* must be provided. Each file system receives the critical threshold value (specified with the `-c` option), warning threshold value (specified with the `-w` option), and enabled state (specified with the `-e` option) or disabled state (specified with the `-d` option) supplied on the command line. (See those options for value ranges.) Only those options specifically included are changed in the specified file system table entries. The special file system name `all` can be used to modify every entry in the table. If you want to disable monitoring for all file systems, use the following command line:

```
fsmon -m -d all
```

It is an error to name a file system that has not been added to the daemon's table (by using the `-a` option). This option cannot be used with the `-a` option.

`-n` *cw* Specifies that no operator message is sent for the specified state. *cw* takes the following values:

`c` Specifies that no message is sent for the critical state.

`w` Specifies that no message is sent for the warning state.

`cw` Specifies that no message is sent for either the critical or warning state.

`-p` *path* Specifies an alternative path name for the files and pipes used for communication between `fsmon` and `fsdaemon`(8). This option is intended for testing and development when an alternative copy of `fsdaemon`(8) that does not interfere with the installed daemon may be useful. See the `fsdaemon`(8) man page for more information on this capability.

`-q` Terminates `fsdaemon`(8). This option may be enabled by an `fsdaemon`(8) option as elected by the system administrator. A message is displayed if this option has not been accepted by the daemon.

`-R` Resets threshold detection on the specified *filesystems*. When `fsdaemon`(8) starts a critical or warning command, it does not initiate another command on the same file system until a reset has been received. This allows the command to execute without the possibility of another command of the same type interfering with it. (The critical command may start while the warning command is running.) When this option is specified, all internal information concerning the specified *filesystems* is discarded, except the enabled or disabled state and the threshold values, and on the next cycle, `fsdaemon`(8) reevaluates these file systems. This command should be used with care because it removes information as to whether commands are running on the affected file systems.

The special file system name `all` causes `fsdaemon`(8) to reset every file system it knows.

`-r` *state* Removes the selected "command running" state from the specified *filesystems*. Possible states are as follows:

`c` Removes "critical command warning" state.

`w` Removes "warning command running" state.

Either or both of these states may be specified, but at least one must appear.

This form of resetting only removes the command running state or states specified and does not discard other state information. This is the recommended way to end a warning or critical command. The special file system name `all` causes `fsdaemon`(8) to remove the specified state from every file system it knows.

`-s` *state* Shows the status of file systems in the specified *state*. The following states are valid:

`c` Critical threshold detected.

`d` Disabled file system entries.

`e` Enabled file system entries.

`w` Warning threshold detected.

Any combination of states may be specified, but at least one must appear. The default is `cdew`.

When `fsmon` is executed with this option, a status display is returned. The status of the specified file systems is indicated in the display's `Status` column. The following list describes the values that appear in each position in the column. Position one always contains `D`, `E`, or `*`. A position that contains a "`-`" means that the status does not apply to the corresponding file system.

`E-------` Monitoring is enabled.

`D-------` Monitoring is disabled.

`*-------` Monitoring error.

`-C------` Critical threshold detected.

`--c-----` Manual critical forced.

`---X----` Critical command executing.

`----W---` Warning threshold detected.

`-----w--` Manual warning forced.

`------X-` Warning command executing.

`-------?` Internal error.

`-t` Enables terse mode. When this option is specified, `fsmon` does not show display headers or any informative messages.

`-w` *nnn* Specifies warning threshold value. This value specifies the percentage of allocated file system space, which, when reached, triggers the execution of the warning command. (The warning command is specified when `fsdaemon`(8) is initiated.) The value range is 0.0 < *nnn* < 100.0; the default value is 85.0%.

filesystems The special file system name `all` represents all known file system names and can be used in any command line as a value for the *filesystems* operand.

NOTES

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

Active Category	**Action**
`system`, `secadm`, `sysadm`	Allowed to use this command.

If the `PRIV_SU` configuration option is enabled, the super user is allowed to use this command.

FILES

`/usr/spool/fsmonitor/Fd.log`	Active log file
`/usr/spool/fsmonitor/Fsdaemon.pid`	Daemon's process ID
`/usr/spool/fsmonitor/Fsdmn.cmd_to`	Daemon command input pipe
`/usr/tmp/Fm_in`*xxxxx*	Daemon command output pipe

SEE ALSO

`fsdaemon`(8)

UNICOS Resource Administration, Cray Research publication SG–2302

NAME

`fsmpptest` – MPP multi-PE application file system and disk device test including performance measurement and data comparison

SYNOPSIS

`mpprun -n 4 fsmpptest` [`-b` *bsize*] [`-e` *esize*] [`-i` *isize*] [`-f` *fsize*] [`-m` *mtfer*] [`-s`] [`-r`] [`-p`] [`-S`] [`-A` *nbufs*] [`-a` *nbufs*] [`-B`] [`-R`] [`-E`] [`-X` *cbits*] [`-x` *cblks*] [`-I` *part*] [`-d`] [`-w`] [`-N`] [`-c`] [`-P` *prefx*] [`-n` *npass*] [`-t` *dtype*] [`-u` *diou*] [`-z`] [`-Z`] [`-Y` *fblks*] [`-M` *MHz*] [`-D` *rlnmx*] [`-C` *comnt*] [`-T` *rftag*] [`-o`] [`-V`] *file1 file2 file3 file4* (for a 4 PE application)

IMPLEMENTATION

Cray MPP systems

DESCRIPTION

`fsmpptest` is a multiple PE, multiple file test for MPPs. `fsmpptest` tests a file system's or disk device's basic functionality, data integrity, and performance. The performance of several items are measured and reported.

By default, file system sequential synchronous write, write - preallocated, and read performance are measured for transfer sizes of 4 KBytes to 32 KBytes. The transfer size is incremented by 4 KBytes at a time. The default file sizes are 3,200 KBytes. Each file gets a different data pattern. Each transfer contains the virtual PE number of the PE doing the I/O and also contains the transfer number.

`fsmpptest` is based on `fstest`(8), which is based on `pddtest`(8). The main difference in options between `fsmpptest` and `fstest` are that `fsmpptest` doesn't support random I/O. Random I/O could be added if needed. Also, `-P` specifies the file name prefix, whereas in `fstest` it specifies the data pattern number.

The following areas are tested and measured by `fsmpptest`:

```
stat
unlink
open
ialloc
write
read
close
```

The following information is reported:

- environment information
- write rates
- write - preallocated rates

- read rates
- other times
- system times
- detailed times

`fsmpptest` puts the information it produces into seven files. They are:

`fst.w.out.rftg`	Writes rates
`fst.wp.out.rftg`	Writes - preallocated rates
`fst.r.out.rftg`	Reads rates
`fst.o.out.rftg`	Other times
`fst.s.out.rftg`	System times
`fst.d.out.rftg`	Detailed times
`fst.t.out.rftg`	Terminal output (`stdout` and `stderr`)

OPTIONS

Transfer Size Options

`-b` *bsize* Beginning transfer size in bytes. KBytes (suffixed with a K), 4 KByte blocks (suffixed with a B), 512 byte blocks (suffixed with a b), MBytes (suffixed with a M), M2Bytes (suffixed with a m), GBytes (suffixed with a G), or G2Bytes (suffixed with a g).

`-e` *esize* Ending transfer size in bytes. KBytes (suffixed with a K), 4 KByte blocks (suffixed with a B), 512 byte blocks (suffixed with a b), MBytes (suffixed with a M), M2Bytes (suffixed with a m), GBytes (suffixed with a G), or G2Bytes (suffixed with a g).

`-i` *isize* Transfer size increment in bytes. KBytes (suffixed with a K), 4 KByte blocks (suffixed with a B), 512 byte blocks (suffixed with a b), MBytes (suffixed with a M), M2Bytes (suffixed with a m), GBytes (suffixed with a G), or G2Bytes (suffixed with a g).

`-f` *fsize* File size which determines the total number of transfers. This can be in bytes; KBytes (suffixed with a K), 4 KByte blocks (suffixed with a B), 512 byte blocks (suffixed with a b), MBytes (suffixed with a M), M2Bytes (suffixed with a m), GBytes (suffixed with a G), or G2Bytes (suffixed with a g).

`-m` *mtfer* If set, the maximum number of transfers to do at each transfer size.

I/O Type Options

`-s` Sequential I/O.

`-r` Random I/O. Not supported.

`-p` Pipe I/O. Not supported.

`-S` Synchronous I/O.

`-A` *nbufs* Asynchronous I/O using recalls specifying the number of buffers to use. Not supported.

`-a` *nbufs* Asynchronous I/O using signals specifying the number of buffers to use. Not supported.

`-B` Library buffered I/O. Not supported.

I/O Options

`-R` Raw I/O `O_RAW | O_WELLFORMED`.

`-E` SFS file exclusive open (`O_SFSXOP | O_SFS_DEFER_TM`).

`-X` *cbits* Preallocation bit mask specifying which partition or partitions to put the file on. This facilitates user striping of a file. This should probably be changed so all files aren't striped on the same partitions.

`-x` *cblks* Number of 4 KByte blocks to allocate per partition for the file. This can be used to preallocate a file or to determine the stripe factor for user striping.

`-I` *part* File preallocation using `ialloc` starting at the specified partition with the partition number being incremented by one for each additional file.

`-d` Disk device I/O (to a `/dev` device). Do I/O to a character or block special file *iofile*. Only reads are done unless `-w` is specified to do writes. When selecting this option, stats, unlinks, and preallocated writes are not done. This option can be selected to do single direction I/O to a disk device or file system. To do just reads use `-d` and to do just writes use `-wN`.

`-w` For disk device I/O, do writes as well as reads.

`-N` For disk device I/O, don't do reads.

Other Options

`-c` Compare data. Compares data written to data read. The first and last words of the transfer contain the virtual PE number of the PE doing the I/O. The second and second to last words of the transfer contain the transfer file position for that transfer. This option will affect read overall transfer rates - use the read "Overall Transfer Rate" following the `dc` read or the "Mean Individual Transfer Rates" in the detailed times report when evaluating performance with this option selected. The file can be evaluated using `od -t o8` *iofile*.

`-P` *prefx* File name prefix. All files are prefixed with this character string.

`-n` *npass* Number of passes through the test.

`-t` *dtype* Disk type to use to determine sizes. This is specified as DD*nnn* or ND*nn*. The disk overall size will be used except if `-f` or `-m` is specified, then one of these will be used. Optimal size transfers (usually cylinder) will be done if `-u` is not specified. Look at `diskspec.h` for the disk types available.

`-u` *diou* Disk I/O unit specifier to use for sizes. Use the first letter of sector, track, cylinder, Parity group, Stripe width, or optimal I/O unit to specify this.

`-z` For DFS - close the file after writing it to sync up the I/O. Only use the "Total Transfer Rates" when evaluating performance with this option selected.

`-Z` For DFS - flush the DFS cache before reading the file. Note that all PEs will do this.

`-Y` *fblks* Flush the system buffer cache before reading the file. If set to 1, 10,000 blocks are flushed per PE. If set to a number greater than 1, this many blocks of cache are flushed per PE. Flushing just the size of the cache instead of the default 10,000 blocks saves time. To determine the size of the system buffer cache and thus how many blocks to flush use `sysconf |grep cache`. If the test is only utilizing one File Server, then divide the cache size by the number of PEs doing I/O to flush the cache as quickly as possible.

`-M` *MHz* CPU clock in MHz - overrides the OS value.

`-D` *rlnmx* If set to 0, turns off the detailed I/O timings report.

`-C` *comnt* Comment describing what's being tested.

`-T` *rftag* Result files name tag.

`-o` Don't print output to the terminal. All output will go to just the `fst.t.out*` file.

`-V` Revision level.

file1 file2 file3 file4
File names to test (for a 4 PE application).

NOTES

In general, for doing performance testing, `fsmpptest` should be run on a system without other activity on it.

Data comparison affects read overall transfer rates - use the read "Total Transfer Rate" following the `dc` read or the "Mean Individual Transfer Rates" in the detailed times report when evaluating performance if data comparison is done. The results including data comparison are noted with a `dc`. Reads with data comparison include the overhead of data comparison.

To print `fsmpptest` result files use:

```
enscript -Br -f Courier8 fsmppt.*.out
```

`fsmpptest` result files can be easily graphed using the Wingz (HSTools) spreadsheet on Sun systems. They can be preprocessed for graphing using the `fsedgraph` command.

`fsmpptest` supports the following size suffixes:

K KByte = 1,024 bytes

B Block = 4,096 bytes

b block = 512 bytes

M MByte = $1{,}000^2$ bytes

m M2Byte = $1{,}024^2$ bytes

G GByte = $1{,}000^3$ bytes

g G2Byte = $1{,}024^3$ bytes

EXAMPLES

Example 1: Do a quick performance test of a file system using a 4 PE test:

```
mpprun -n 4 fsmpptest /fs/file1 /fs/file2 /fs/file3 /fs/file4
```

Example 2: Do a quick performance test of a file system using a 4 PE test with data comparison and prefixing the file names with /*tmp*/:

```
mpprun -n 4 fsmpptest -c -P /tmp/ f1 f2 f3 f4
```

Example 3: Do a four file performance test making sure not to overflow the system buffer cache by specifying a file size maximum of 250 blocks:

```
mpprun -n 4 fsmpptest -f 250B -P /tmp/ f1 f2 f3 f4
```

Example 4: Test a file system using a 4 PE test doing raw I/O with transfers ranging from one sector to 32 KBytes against file sizes of 3200 KBytes doing sequential synchronous I/O. Also put a comment at the top of each report file explaining what was tested and tag the report file names with a time stamp. In addition, prefix the files with the file system.

```
mpprun -n 4 fsmpptest -R -b 4096 -e 32768 -i 4096 -f 3276800 -s -S x
-C "Cray File System Raw I/O Test" -T ".‘date +%H%M‘" x
-P /tmp/ f1 f2 f3 f4
```

Example 5: Test a file system using an 8 PE test doing raw I/O with transfers ranging from 1/4 M2Byte to 1 M2Byte against file sizes of 100 M2Bytes doing sequential synchronous I/O. Test only 100 transfers at each transfer size. Also put a comment at the top of each report file explaining what was tested and tag the report file names. In addition, prefix the files with the file system.

```
mpprun -n 8 fsmpptest -R -b 256K -e 1m -i 256K -f 100m -m 100 x
-C "Large File Raw I/O Test" -T ".lrg" x
-P /fs/ f1 f2 f3 f4 f5 f6 f7 f8
```

Example 6: Test a file system using a 4 PE test doing raw I/O with transfers ranging from 32 KBytes to 256 KBytes against file sizes of 25600 KBytes doing sequential synchronous I/O. Also put a comment at the top of each report file explaining what was tested and tag the report file names. In addition, prefix the files with the file system.

```
mpprun -n 4 fsmpptest -R -b 32K -e 256K -i 32K -f 25600K x
-C "Medium File Raw I/O Test" -T ".med" x
-P /fs/ f1 f2 f3 f4
```

Example 7: Test the performance of a file system using a 4 PE test doing raw one M2Byte transfers:

```
mpprun -n 4 fsmpptest -R -b 1m -e 1m -f 100m x
-C "One MByte Raw Transfers Test" -T ".1mb" x
-P /fs/ f1 f2 f3 f4
```

Example 8: Do a quick performance test of four disk partitions including writes and data comparison using a 4 PE test - note that this test will over-write data on these disk partitions:

```
mpprun -n 4 fsmpptest -dwc -P "/dev/xdd/" dd314.0 dd314.1 dd314.2 dd314.3
```

Example 9: Do a quick performance test of four disk partitions at the block device level including writes using a 4 PE test - note that this test will over-write data on these disk partitions:

```
mpprun -n 4 fsmpptest -dw -P /dev/dsk/ dd314.0 dd314.1 dd314.2 dd314.3
```

Example 10: Run `fsmpptest` over night comparing the data using a 16 PE test:

```
mpprun -n 16 fsmpptest -c -n 1000000 -P /fs/ f1 f2 f3 f4 f5 f6 f7 f8 \
f9 f10 f11 f12 f13 f14 f15 f16
```

Example 11: Test a file system using a 2 PE test and printing no output to the terminal:

```
mpprun -n 2 fsmpptest -c -n 1000000 -T .2pes -o /fs/file1 /fs/file2 &
```

Example 12: Test four whole DA-301s using track transfers - note that this test will over-write data on these disks:

```
mpprun -n 4 fsmpptest -dwc -t DA301 -u t x
-C "Four DA-301 Disk Devices Test" -T ".da301s" x
-P /dev/pdd/ da301.0.all da301.1.all da301.2.all da301.3.all
```

Example 13: Test four new DD-314s in a manner similar to `pddtest` using a 4 PE test:

```
mpprun -n 4 fsmpptest -dwc -t DD314 -n 1000000 x
-P /dev/xdd/ dd314.0 dd314.1 dd314.2 dd314.3
```

Example 14: Run `fsmpptest` over night comparing the data to test 100,000 block slices of DD-314s including writes using transfer sizes ranging from one block to one track using a 4 PE test:

```
mpprun -n 4 fsmpptest -dwc -b 1B -e 17B -i 1B -f 100000B -n 1000000 n
-C "DD-314s Test" -T ".‘date +%H%M‘" n
-P /dev/xdd/ dd314.0 dd314.1 dd314.2 dd314.3
```

Example 15: Test a DFS file system using 4 files and a 4 PE test:

```
mpprun -n 4 fsmpptest -T .dfs -P "/:/cray/cool/ptmp/" f1 f2 f3 f4
```

Example 16: Test a DFS file system closing the file after writing it to sync up the I/O using a 4 PE test. Only use the "Total Transfer Rates" when evaluating the reports in this case.

```
mpprun -n 4 fsmpptest -z -T .dfsz -P "/:/cray/cool/ptmp/" f1 f2 f3 f4
```

Example 17: Test a DFS file system flushing the system buffer cache and the DFS cache before reading the file using a 4 PE test:

```
mpprun -n 4 fsmpptest -Y 500 -Z -T .dfsYZ -P "/:/cray/cool/ptmp/" f1 f2 f3 f4
```

Example 18: Test a NFS file system using an 8 PE test:

```
mpprun -n 8 fsmpptest -T .nfs -P "/cray/cool/ptmp/" f1 f2 f3 f4 f5 f6 f7 f8
```

Example 19: Test a NFS file system flushing the 2,000 block system buffer cache before reading the files using a 4 PE test:

```
mpprun -n 4 fsmpptest -Y 500 -T .nfsY -P /cray/frost/tmp/ f1 f2 f3 f4
```

SEE ALSO

`fstest`(8)

NAME

`fsoffload` – Lists files and directories on a logical device

SYNOPSIS

`/etc/fsoffload` [`-a`] [`-c`] [`-d`] [`-D`] [`-f` *pathname*] [`-h`] [`-p` *pdev*] [`-r`] [`-v`] *ldev*
`/etc/fsoffload` [`-a`] [`-c`] [`-d`] [`-D`] [`-f` *pathname*] [`-h`] [`-m`] [`-M`] [`-r`] [`-s` *slice*[`,`*slice*`,...`]] [`-v`] *ldev*

IMPLEMENTATION

Cray PVP systems

DESCRIPTION

The `fsoffload` command lists files and directories that are resident or partially resident on particular slices of a logical device. A file is considered resident if its inode or any of its data reside on one of the specified logical device components.

The `fsoffload` command accepts the following options and operand:

`-a` Lists all files, directories, and special files found residing on the specified logical device components. The `-r` and `-d` options do not have to be specified with this option, because it performs their functions.

`-c` Lists all children and descendants of directories that are resident on the specified logical device components.

`-d` Lists all directories on the specified logical device components.

`-D` Debug mode; prints additional information about file residency.

`-f` *pathname*
Specifies a file or directory to be searched for files residing on the specified logical device components. If this option is not specified, the mount point found in the kernel mount table is assumed.

`-h` Prints help message.

`-m` Rebuilds files. This option causes files resident on the specified slices to be copied (to their original names). If the specified slices are on a read-only disk, the files are moved off the disk. This option alone does not list the files that are copied; if you wish a list of the files, use this option in conjunction with an option that provides a list, such as `-a`.

`-M` Rebuilds directories. This option causes directories resident on the specified slice(s) to be rebuilt. If the specified slices are on a read-only disk, the directories are moved off the disk.

`-p` *pdev* Specifies a physical device; *pdev* should have a component composing part of the logical device. *pdev* specifies a component name as found in the superblock. You cannot use this option with the `-s` option.

`-r` Lists all regular files on the specified logical device components.

`-s` *slice*[`,` *slice*`,` ...]
Specifies a number representing a particular *slice* of a logical device, as represented in the superblock. The slice value is a base-10 slice number. You can specify multiple values, separated by commas. This option cannot be used with the `-p` option.

`-v` Displays the components of the specified logical device, showing which components are marked for residency search.

ldev A logical device; *ldev* must be a mounted file system. This operand is required.

NOTES

This command can be used to observe sensitive data on a device. Only appropriately authorized users can use this command.

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

Privilege Text	Action
`showall`	Allowed to use this command.

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

Active Category	Action
`system`, `secadm`, `sysadm`	Allowed to use this command.

If the `PRIV_SU` configuration option is enabled, the super user is allowed to use this command.

SEE ALSO

`pddconf`(8), `pddstat`(8)

General UNICOS System Administration, Cray Research publication SG–2301

NAME

fstest – Tests file systems and disk devices

SYNOPSIS

fstest [-b *bsize*] [-e *esize*] [-i *isize*] [-f *fsize*] [-m *mtfer*] [-s] [-r] [-p] [-S] [-A *nbufs*] [-a *nbufs*] [-B] [-R] [-E] [-K] [-X *cbits*] [-x *cblks*] [-I *part*] [-d] [-w] [-N] [-c] [-P *pattn*] [-n *npass*] [-t *dtype*] [-u *diou*] [-v] [-z] [-Z] [-Y *fblks*] [-M *MHz*] [-D *rlnmx*] [-C *comnt*] [-T *rftag*] [-o] [-V] *iofile*

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

fstest tests a file system's or disk device's basic functionality, data integrity, and performance. The performance of several items are measured and reported. fstest is a single process, single file test.

By default, file system sequential synchronous write, write - preallocated, and read performance are measured for transfer sizes of 4 KBytes to 32 KBytes. The transfer size is incremented by 4 KBytes at a time. The default file size is 3,200 KBytes. The default data pattern is data pattern 0.

fstest is based on pddtest(8). fstest is similar to pddtest with timers that can also be used at the file system level. It has different defaults and command line options, however. The main item missing from fstest which pddtest has is run on error. This could be added if needed. Other differences are fstest can do random I/O without doing sequential I/O first and can also test a range of transfer sizes. fstest numbers each transfer instead of numbering each sector like pddtest.

The following areas are tested and measured by fstest:

```
stat
unlink
open
ialloc
write
read
close
```

The following information is reported:

- environment information
- write rates
- write - preallocated rates
- read rates
- other times

- system times
- detailed times (optional)

`fstest` puts the information it produces into seven files. They are:

`fst.w.out.rftg`	Writes rates
`fst.wp.out.rftg`	Writes - preallocated rates
`fst.r.out.rftg`	Reads rates
`fst.o.out.rftg`	Other times
`fst.s.out.rftg`	System times
`fst.d.out.rftg`	Detailed times (optional)
`fst.t.out.rftg`	Terminal output (`stdout` and `stderr`)

OPTIONS

Transfer Size Options

`-b` *bsize* Beginning transfer size in bytes. KBytes (suffixed with a K), 4 KByte blocks (suffixed with a B), 512 byte blocks (suffixed with a b), MBytes (suffixed with a M), M2Bytes (suffixed with a m), GBytes (suffixed with a G), or G2Bytes (suffixed with a g).

`-e` *esize* Ending transfer size in bytes. KBytes (suffixed with a K), 4 KByte blocks (suffixed with a B), 512 byte blocks (suffixed with a b), MBytes (suffixed with a M), M2Bytes (suffixed with a m), GBytes (suffixed with a G), or G2Bytes (suffixed with a g).

`-i` *isize* Transfer size increment in bytes. KBytes (suffixed with a K), 4 KByte blocks (suffixed with a B), 512 byte blocks (suffixed with a b), MBytes (suffixed with a M), M2Bytes (suffixed with a m), GBytes (suffixed with a G), or G2Bytes (suffixed with a g).

`-f` *fsize* File size which determines the total number of transfers. This can be in bytes; KBytes (suffixed with a K), 4 KByte blocks (suffixed with a B), 512 byte blocks (suffixed with a b), MBytes (suffixed with a M), M2Bytes (suffixed with a m), GBytes (suffixed with a G), or G2Bytes (suffixed with a g).

`-m` *mtfer* If set, the maximum number of transfers to do at each transfer size.

I/O Type Options

`-s` Sequential I/O.

`-r` Random I/O. For random I/O, the file is only created once.

`-p` Pipe I/O. Not supported.

`-S` Synchronous I/O.

`-A` *nbufs* Asynchronous I/O using recalls specifying the number of buffers to use. Not supported.

`-a` *nbufs* Asynchronous I/O using signals specifying the number of buffers to use. Not supported.

`-B` Library buffered I/O. Not supported.

I/O Options

`-R` Raw I/O (`O_RAW | O_WELLFORMED`).

`-E` SFS file exclusive open (`O_SFSXOP | O_SFS_DEFER_TM`).

`-X` *cbits* Preallocation bit mask specifying which partition or partitions to put the file on. This facilitates user striping of a file.

`-x` *cblks* Number of 4 KByte blocks to allocate per partition for the file. This can be used to preallocate a file or to determine the stripe factor for user striping.

`-I` *part* Preallocate the file on the specified partition using `ialloc`.

`-d` Disk device I/O (to a `/dev` device). Do I/O to a character or block special file *iofile*. Only reads are done unless `-w` is specified to do writes. When selecting this option, `stat`s, `unlink`s, and preallocated writes are not done. This option can be selected to do single direction I/O to a disk device or file system. To do read operations only use `-d` and to do write operations only use `-wN`.

`-w` For disk device I/O, do writes as well as reads.

`-N` For disk device I/O, don't do reads.

Other Options

`-c` Compare data. Compares data written to data read. The first and last words of the transfer contain the transfer file position for that transfer. This option will affect read overall transfer rates - use the read "Overall Transfer Rate" following the `dc` read or the mean from the "Individual Transfer Rates" when evaluating performance with this option selected. The file can be evaluated using `od -t o8` *iofile*.

`-P` *pattn* Data pattern number to use. This is a number from 0 - 9 since there are 10 different data patterns to choose from.

`-n` *npass* Number of passes through the test. For each pass through the test, the data pattern is incremented by one.

`-t` *dtype* Disk type to use to determine sizes. This is specified as DD*nnn* or ND*nn*. The disk overall size will be used except if `-f` or `-m` is specified, then one of these will be used. Optimal size transfers (usually cylinder) will be done if `-u` is not specified. Look at `diskspec.h` for the disk types available.

`-u` *diou* Disk I/O unit specifier to use for sizes. Use the first letter of sector, track, cylinder, Parity group, Stripe width, or optimal I/O unit to specify this.

`-v` Vary I/O - vary between sequential and random I/O. Sequential I/O is done one pass and then random I/O is done the next pass.

`-z` For DFS - close the file after writing it to sync up the I/O. Only use the "Overall Transfer Rates" when evaluating performance with this option selected.

`-Z`	For DFS - flush the DFS cache before reading the file.
`-Y` *fblks*	Flush the system buffer cache before reading the file. If set to 1, 10,000 blocks are flushed. If set to a number greater than 1, this many blocks of cache are flushed. Flushing just the size of the cache instead of the default 10,000 blocks saves time. To determine the size of the system buffer cache and thus how many blocks to flush use `sysconf |grep NBUF`.
`-M` *MHz*	CPU clock in MHz - overrides the OS value.
`-D` *rlnmx*	Detailed I/O timings report. If set to 1, all individual transfer times are reported. If set to a number greater than 1, it reports this many lines worth of individual transfer times for each operation.
`-C` *comnt*	Comment describing what's being tested.
`-T` *rftag*	Result files name tag.
`-o`	Don't print output to the terminal. All output will go to just the `fst.t.out*` file.
`-V`	Revision level.
iofile	File name to test.

NOTES

In general, for doing performance testing, `fstest` should be run on a system without other activity on it. When testing the random I/O performance of a disk, the entire disk should be used for the test.

Data comparison affects read overall transfer rates - use the read "Overall Transfer Rate" following the `dc` read or the mean from the "Individual Transfer Rates" when evaluating performance if data comparison is done. The results including data comparison are noted with a `dc`. Reads with data comparison include the overhead of data comparison.

To print `fstest` result files use:

```
enscript -Br -f Courier8 fst.*.out
```

`fstest` result files can be easily graphed using the Wingz (HSTools) spreadsheet on Sun systems. They can be preprocessed for graphing using the `fsedgraph` command.

`fstest` supports the following size suffixes:

K	KByte = 1,024 bytes
B	Block = 4,096 bytes
b	block = 512 bytes
M	MByte = $1{,}000^2$ bytes
m	M2Byte = $1{,}024^2$ bytes
G	GByte = $1{,}000^3$ bytes
g	G2Byte = $1{,}024^3$ bytes

EXAMPLES

Example 1: Do a quick performance test of a file system:

```
fstest /fs/iofile
```

Example 2: Do a quick performance test of a file system with data comparison:

```
fstest -c /tmp/iofile
```

Example 3: Test a file system with transfers ranging from one sector to 32 KBytes against a file size of 3200 KBytes doing sequential synchronous I/O. Also report the first 10 lines of individual I/O timings for each operation and put a comment at the top of each report file explaining what was tested. In addition, tag the report file names with a time stamp.

```
timex fstest -b 4096 -e 32768 -i 4096 -f 3276800 -s -S -D 10 x
-C "Cray File System Test" -T ".‘date +%H%M‘" x
/tmp/iofile
```

Example 4: Test a file system with transfers ranging from 1/4 M2Byte to 1 M2Byte against a file size of 100 M2Bytes doing sequential synchronous I/O. Test only 100 transfers at each transfer size. Also report the first 10 lines of individual I/O timings for each operation and put a comment at the top of each report file explaining what was tested. In addition, tag the report file names.

```
fstest -b 256K -e 1m -i 256K -f 100m -m 100 -s -S -D 10 x
-C "Large File System Test" -T ".lrg" x
/fs/iofile
```

Example 5: Test a file system with transfers ranging from 32 KBytes to 256 KBytes against a file size of 25600 KBytes doing sequential synchronous I/O. Also report the first 10 lines of individual I/O timings for each operation and put a comment at the top of each report file explaining what was tested. In addition, tag the report file names.

```
fstest -b 32K -e 256K -i 32K -f 25600K -s -S -D 10 x
-C "Medium File System Test" -T ".med" x
/fs/iofile
```

Example 6: Test the performance of a file system using one M2Byte transfers:

```
fstest -b 1m -e 1m -f 100m -s -S -D 10 x
-C "One MByte Transfers File System Test" -T ".1mb" x
/fs/iofile
```

Example 7: Do random I/O to a disk in a file system:

```
fstest -r /fs/iofile
```

Example 8: Test the raw I/O performance of a file system:

```
fstest -R -b 4B -e 32B -i 4B -f 3200B x
-C "Raw I/O Test" -T ".raw" /tmp/iofile
```

Example 9: Test the raw I/O performance of a file striped across 4 file system partitions using a stripe factor of a track (48 blocks for this example). Check which file system partitions are available by doing `df -p /fs`. For this example the file system consists of 5 partitions, 0 - 4. Use partitions 1 - 4 of the file system for this test. The bit mask to select these partitions is determined as follows:

$2^1 + 2^2 + 2^3 + 2^4 = 30$

The bit mask could also have been determined by using:

$(2^5 - 1) - 2^0$

Do transfers of 4 tracks (192 blocks) to a 400 track file.

```
fstest -R -X 30 -x 48 -b 192B -e 192B -f 19200B /fs/iofile
```

Example 10: Do random raw I/O to a DD-302 in a file system. Only do 1,000 random transfers at each transfer size. To determine the file size use df to determine the amount of free space.

```
fstest -rR -f 395920B -m 1000 x
-C "DD-302 File System Random Raw I/O Test" -T ".ranraw" x
/fs/iofile
```

Example 11: Do a quick performance test of a disk partition including writes and data comparison - note that this test will over-write data on this disk partition:

```
fstest -dwc /dev/pdd/dd302
```

Example 12: Do a quick performance test of a disk partition at the block device level including writes - note that this test will over-write data on this disk partition:

```
fstest -dw /dev/dsk/dd302
```

Example 13: Run `fstest` over night comparing the data and varying the I/O between sequential and random to test a file system (the data pattern will also vary):

```
fstest -c -v -n 1000000 /fs/iofile
```

Example 14: Test a file system using two processes with each using a different data pattern and printing no output to the terminal:

```
fstest -c -n 1000000 -T .p1 -o /fs/iofile &
fstest -c -P 1 -n 1000000 -T .p2 -o /fs/iofile2 &
```

Example 15: Test a whole DA-301 using track transfers - note that this test will overwrite data on this disk:

```
fstest -dwc -t DA301 -u t x
-C "DA-301 Disk Device Test" -T ".da301" x
/dev/pdd/da301.all
```

Example 16: Test a new DD-302 as you would using `pddtest`:

```
fstest -dwcv -t DD302 -n 1000000 /dev/pdd/dd302
```

Example 17: Test the performance of a four disk file system using four files. Only write the files once so writes don't dominate the test. Use a large enough file size so the test runs long enough to get meaningful results. Evaluate the run using the mean of the "timex real time" results or the "`fst.s.out` pass total elapsed time" results.

Put the following in a script and run it:

```
# four file test using a transfer size of one block

TAG=".‘date +%H%M%S‘"

timex fstest -dw -b 4096 -e 4096 -f 4096000 -s -S x
-C "Four-File File System Test - File 1" -T ".1$TAG" -o x
/fs/iofile1 &

timex fstest -dw -b 4096 -e 4096 -f 4096000 -s -S x
-C "Four-File File System Test - File 2" -T ".2$TAG" -o x
/fs/iofile2 &

timex fstest -dw -b 4096 -e 4096 -f 4096000 -s -S x
-C "Four-File File System Test - File 3" -T ".3$TAG" -o x
/fs/iofile3 &

timex fstest -dw -b 4096 -e 4096 -f 4096000 -s -S x
-C "Four-File File System Test - File 4" -T ".4$TAG" -o x
/fs/iofile4 &

wait

# end of four file test
```

Example 18: Run `fstest` overnight, comparing the data and varying the I/O between sequential and random to test a DD-314 including writes (the data pattern will also vary and cylinder transfers will be done):

```
fstest -dwcv -t DD314 -n 1000000 /dev/xdd/dd314
```

Example 19: Run `fstest` overnight, comparing the data and varying the I/O to test a 100,000 block slice of a DD-314 including writes using transfer sizes ranging from one block to one track:

```
fstest -dwcv -b 1B -e 17B -i 1B -f 100000B -n 1000000 x
-C "DD-314 Test" -T ".‘date +%H%M‘" /dev/xdd/dd314
```

Example 20: Test a DD-314 including writes doing track random I/O and data comparison:

```
fstest -dwc -r -t DD314 -u t /dev/xdd/dd314
```

Example 21: Test a DFS file system:

`fstest -T .dfs /:/cray/cool/`*`ptmp/iofile`*

Example 22: Test a DFS file system closing the file after writing it to sync up the I/O. Only use the "Overall Transfer Rates" when evaluating the reports in this case.

`fstest -z -T .dfsz /:/cray/cool/`*`ptmp/iofile`*

Example 23: Test a DFS file system flushing the system buffer cache and the DFS cache before reading the file:

`fstest -Y 2000 -Z -T .dfsYZ /:/cray/cool/`*`ptmp/iofile`*

Example 24: Test a NFS file system:

`fstest -T .nfs /cray/cool/`*`ptmp/iofile`*

Example 25: Test a NFS file system flushing the system buffer cache before reading the file:

`fstest -Y 2000 -T .nfsY /cray/frost/`*`tmp/iofile`*

Example 26: Test the raw I/O performance of a preallocated file on partition 1 of a file system

`fstest -R -I 1 /`*`fs/iofile`*

Example 27: Test a ND-40 including writes doing data comparison and varying the I/O using optimal I/O transfers:

```
fstest -dwcv -t ND40_64KR5 /dev/hdd/nd40.1.h1.b1
```

Example 28: Test a ND-40 in a file system doing raw I/O, data comparison, varying the I/O between sequential and random, using optimal size I/O transfers, and preallocating the file on partition 0 (to determine the file size use df to determine the amount of free space):

`fstest -Rcv -t ND40_64KR5 -f 23034352B -I 0 /nd40/`*`iofile`*

Example 29: Do a quick test of some new ND-40 nodes to make sure they work - note that this test will overwrite data on these disk slices:

```
fstest -dw -t ND40R_R5 -f 10000B /dev/hdd/nd40.p11.f16.1
fstest -dw -t ND40R_R1 -f 1000B /dev/hdd/nd40.p21.f32.1
fstest -dw -t ND40R_R1 -f 1000B /dev/dsk/nd40test
```

SEE ALSO

fsmpptest(8)

NAME

ftpd – Invokes the Internet file transfer protocol server

SYNOPSIS

/etc/ftpd [-d] [-h] [-k] [-l] [-v] [-r[0|1]] [-R] [-s *shift*] [-t *timeout*] [-T *seconds*] [-u*mask*] [-Sc *tos*] [-Sd *tos*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The ftpd command invokes the Internet file transfer protocol server process. The server uses transmission control protocol (TCP) and listens at the port specified in the ftp service specification; see services(5).

The ftpd command invokes the centralized identification and authorization library routines to validate the user ID and password.

The ftpd command accepts the following options:

-d Logs debugging information to syslog. (Identical to the -v option.)

-h Disables printing of host-specific information before the user is validated.

-k Allows unknown hosts (for example, those not found in the host's name database) to access Cray Research systems through ftp when security is enabled. By default, hosts that do not have an entry in the host's name database cannot transfer into a secure UNICOS system.

-l Logs general information to syslogd(8).

-v Logs debugging information to syslog. (Identical to the -d option.)

-r[0|1] Enables or disables raw I/O to files. If you specify -r0, raw I/O is disabled; if you specify -r or -r1, it is enabled. If you omit the -r option, raw I/O is enabled by default.

-R Allows a user to send a file to any command, rather than to just a disk file, and it allows the output of any command to be retrieved. You should not enable this option unless you fully understand the ramifications of it; usually, users are allowed only to send and retrieve files. Because this option allows them to execute commands, you might be extending more privileges than you intend.

-s *shift* Sets the default value for the TCP window scale option. If *shift* is on, it is enabled, with a value of 4. If you omit the -s option, this is the default mode. If *shift* is off, it disables the TCP window scale option. If *shift* is a value between 0 and 14, it enables the TCP window scale option with that value.

-t *timeout*
Sets the inactivity time-out period to *timeout* seconds. The default is 15 minutes.

`-T` *seconds* Sets the maximum number of seconds to which the inactivity time-out period can be set. The default is 7200 seconds (2 hours).

`-u`*mask* Sets the default umask to *mask*. The default is 027.

`-Sc` *tos* Sets the IP type-of-service option for the FTP control connection to the value *tos*, which can be a numeric TOS value or a symbolic TOS name found in the `/etc/iptos` file.

`-Sd` *tos* Sets the IP type-of-service option for the FTP data connection to the value *tos*, which can be a numeric TOS value or a symbolic TOS name found in the `/etc/iptos` file.

`-S` *tos* Sets the IP type-of-service option for both the FTP control connection and the FTP data connection to the value *tos*, which can be a numeric TOS value or a symbolic TOS name found in the `/etc/iptos` file.

Currently, the `ftp` server supports the following `ftp` requests (case is not distinguished):

Request	**Description**
`ABOR`	Aborts the previous command (vacuously).
`ADAT`	Base 64 encoded authentication data.
`ALLO`	Allocates storage (vacuously).
`APPE`	Appends to a file.
`AUTH`	Identifies a supported authentication mechanism.
`CDUP`	Changes to parent of current working directory.
`CWD`	Changes the working directory.
`DELE`	Deletes a file.
`ENC`	Privacy protected command. The argument field is a base 64 encoded Telnet string. The server decodes the string and verifies its integrity. The resulting string is interpreted as an FTP command.
`HELP`	Gives help information.
`LIST`	Lists files in a directory (`ls -lg`).
`MIC`	Integrity protected command. The argument field is a base 64 encoded Telnet string. The server decodes the string and verifies its integrity. The resulting string is interpreted as an FTP command.
`MKD`	Makes a directory.
`MODE`	Specifies the data transfer *mode*.
`NLST`	Gives a name list of files in a directory (`ls`).
`NOOP`	Does nothing.
`PASS`	Specifies a password.

PASV	Prepares for server-to-server transfer.
PBSZ	Maximum number of bytes in buffer for protected file transfer.
PORT	Specifies a data connection port.
PROT	Protection level.
PWD	Prints current working directory.
QUIT	Terminates the session.
REST	Restarts the command.
RETR	Retrieves a file.
RMD	Removes a directory.
RNFR	Specifies the *rename-from* file name.
RNTO	Specifies the *rename-to* file name.
SITE CHMOD	Changes the mode on a file.
SITE COPYBUF	Sets copy buffer size.
SITE FULLBUF	Toggles use of full buffers on writes to disk. By default, the buffer is filled before writing to disk begins.
SITE HELP	Gives help for the SITE commands.
SITE IDLE	Gets and sets the inactivity time-out period.
SITE RAWBUF	Toggles whether raw I/O will be used.
SITE SHOWBUF	Shows buffer sizes on transfers.
SITE SOCKBUF	Sets socket buffer size on data socket.
SITE UMASK	Gets and sets the umask value.
SITE WINSHIFT	Gets the status of the use of the TCP window scale option. It can also be used to enable or disable the option. By default, it is enabled with a value of 4.
STAT	Displays status of connection to a file.
STOR	Stores a file.
STOU	Stores a file and gives it a unique name.
STRU	Specifies the data transfer *structure*.
SYST	Gives operating systems information.
TYPE	Specifies the data transfer *type*.
USER	Specifies the user name.
XCUP	Changes to the parent of the current working directory.

XCWD	Changes the working directory.
XMKD	Makes a directory.
XPWD	Prints the current working directory.
XRMD1	Removes a directory.

The remaining ftp requests specified in Internet RFC 959 are recognized but not implemented. The following ftp requests are not specified in RFC 959:

Unimplemented, obsolete:	MAIL, MLFL, MRCP, MRSQ, MSAM, MSND, and MSOM
Implemented, obsolete:	XCUP, XCWD, XMKD, XPWD, and XRMD
Implemented, forthcoming:	MDTM and SIZE

The ftpd command interprets file names the same way the glob command of ftp(1B) does. This lets you use the *, ?, [,], {, }, and ~ metacharacters.

The ftpd command authenticates users according to the following four rules:

1. The user name must be in the /etc/udb file (see udb(5)) and must have a password (that is, it must not have a null password). In this case, the client must provide a password before any file operations can be performed.
2. The user name must not appear in the /etc/ftpusers file (see ftpusers(5)).
3. The user must have a standard shell (see shells(5)).
4. If the user name is anonymous or ftp, an anonymous ftp account must be present in the /etc/udb file (user ftp). In this case, the user is allowed to log in by specifying any password (by convention, this is given as the user's name).

In the last case, ftpd takes special measures to restrict the client's access privileges. The server performs a chroot(2) system call to the home directory of user ftp. To avoid breaching system security, you must construct the ftp directory with care. The following rules are recommended:

~ftp	The ftp home directory should be owned by root and unwritable by anyone.
~ftp/bin	This directory should be owned by the super user and unwritable by anyone. The ls(1) program must be present to support the list commands. This program should have mode 111.
~ftp/etc	This directory should be owned by the super user and unwritable by anyone. The udb(5) file must be present for the ls command to work properly. This file should be mode 444.
~ftp/pub	This directory should have mode 777 and be owned by ftp. You should then place files that are accessible to the anonymous account in this directory.

NOTES

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

Active Category	Action
`system`, `secadm`, `sysadm`	Allowed to use this command.

If the `PRIV_SU` configuration option is enabled, the super user is allowed to use this command.

`ftpd` login requests are recorded in the audit log.

BUGS

The `anonymous` account is a security risk and should be avoided.

The server must run with appropriate privilege to create sockets with privileged port numbers. (A *socket* is a bidirectional structure within the host that sends and receives packets.) It maintains an effective user ID of the logged-in user and uses nonprivileged port numbers for the data connection.

FILES

`/etc/config/spnet.conf`	File that contains a list of individuals and groups that are allowed access to the UNICOS system from a given host or workstation and the services allowed at that workstation. It also contains the minimum and maximum security levels and the minimum and maximum compartments for the client host and workstation.
`/etc/ftpusers`	File that contains the names of users who are denied access to `ftp` from a remote host
`/etc/inetd.conf`	Default configuration file for the `inetd` daemon
`/etc/shells`	File that contains a list of shells that are associated with user accounts
`$HOME/.netrc`	User-created file that consists of the `ftp` user authentication table

SEE ALSO

spnet(8), syslogd(8)

ftp(1B), login(1), privtext(1) in the *UNICOS User Commands Reference Manual*, Cray Research publication SR–2011

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

ia_failure(3C), ia_mlsuser(3C), ia_success(3C), ia_user(3C) in the *UNICOS System Libraries Reference Manual*, Cray Research publication SR–2080

ftpusers(5), group(5), netrc(5), services(5), shells(5), udb(5) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

RFC 959 *File Transfer Protocol (FTP)*

UNICOS Networking Facilities Administrator's Guide, Cray Research publication SG–2304

NAME

`fuser` – Identifies processes using a file or file structure

SYNOPSIS

`/etc/fuser` [`-k`] [`-m`] [`-u`] [`-c`] *files* [[`-`] [`-k`] [`-u`] [`-c`] *files*]
`/etc/fuser` [`-s`] [`-k`] [`-u`] [`-c`] [*major*] *minor* [[`-`] [`-k`] [`-u`] [`-c`] [*major*] *minor*]

IMPLEMENTATION

Cray PVP systems

DESCRIPTION

The `fuser` command lists the process IDs of the processes using the *files* specified on the command line. For block special devices, all processes using any file on that device are listed. The process ID is followed by `c`, `p`, `r`, or `t` character, to specify how the process uses the file. The character `c` signifies that the process is using the file as its current directory; `p` signifies that the process is using the file as the parent of its current directory (only when in use by the system); `r` signifies that the process is using the file as its root directory; and `t` signifies that the process is using the file as its controlling tty.

When used with the `-s` option, `fuser` lists the process IDs of the processes using the devices with minor device number *minor* and major device number *major*. If *major* is omitted, the major device number for sockets is assumed.

The `fuser` command accepts the following options:

`-k` Sends the `SIGKILL` signal to each process. Only the super user can terminate another user's process (see `kill`(2)).

`-m` Interprets *files* as a mounted file system. This is equivalent to replacing *files* with the block device in which *files* appears.

`-u` Prints the login name, in parentheses, following the process ID.

`-c` Prints the command name, in square brackets, following the process ID.

`-` Respecifies options between groups of files on a single command line. If you specify a single hyphen, the new set of options replaces the old set, canceling any options currently in force.

`-s` Lists the process IDs of the processes using the devices with minor device number *minor* and major device number *major*. If *major* is omitted, the major device number for sockets is assumed.

The process IDs are printed as a single line on the standard output, separated by spaces and terminated with a single new line. All other output is written on standard error.

NOTES

Output from `fuser` is restricted to processes running at a security label that the calling user dominates.

If this command is installed with the default privilege assignment list (PAL), a user with the `showall` privilege text is not subject to output restrictions.

EXAMPLES

Example 1: If typed by a super user, the following example terminates all processes that are preventing `/dev/dsk/usr` from being unmounted; it lists the process ID and login name of each process as it is killed.

```
fuser -ku /dev/dsk/usr
```

Example 2: The following example lists process IDs and login names of processes that have the password file open.

```
fuser -u /etc/passwd
```

Example 3: The following example performs both of the preceding operations with a single command line.

```
fuser -ku /dev/dsk/usr - -u /etc/passwd
```

Example 4: The following example lists process IDs, command names, and login names of processes that have socket 37 open.

```
fuser -s -cu 37
```

FILES

`/unicos`	System namelist
`/dev/kmem`	System image

SEE ALSO

`mount`(8)

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

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

General UNICOS System Administration, Cray Research publication SG–2301

NAME

fwtmp, wtmpfix – Manipulates connect accounting records

SYNOPSIS

```
/usr/lib/acct/fwtmp [-c] [-i]
/usr/lib/acct/wtmpfix [files]
```

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

Without arguments, fwtmp reads from the standard input and writes to the standard output, converting binary records of the type found in wtmp to formatted ASCII records. The ASCII version lets you edit records that contain data that is not valid (using vi(1)), or generally maintain the file.

The fwtmp command accepts the following options:

-c Specifies that the output is in binary form. The default is ASCII form.

-i Specifies that the input is in ASCII form. The default is binary form.

files Files that contain connect accounting records.

The wtmpfix command examines the standard input or specified files in wtmp format, corrects the time and date stamps to make the entries consistent, and writes to the standard output. You can use a - in place of *files* to indicate the standard input. If time and date corrections are not performed, acctcon1 (see acctcon(8)) exits with an error code when it encounters certain date-change records.

Each time the date is set, a pair of date-change records are written to /etc/wtmp. The first record is the old date denoted by the string old time placed in the line field and the flag OLD_TIME placed in the type field of the utmp.h structure (defined in the include file utmp.h). The second record specifies the new date and is denoted by the string new time placed in the line field and the NEW_TIME flag placed in the type field. wtmpfix uses these records to synchronize all time stamps in the file.

In addition to correcting time/date stamps, wtmpfix checks the validity of the name field to ensure that it consists only of alphanumeric characters or spaces. If it encounters a name that is considered not valid, it changes the login name to INVALID and writes a diagnostic message to the standard error. In this way, wtmpfix reduces the chance that acctcon1 will fail when processing connect accounting records.

FILES

/etc/wtmp	Login records format file
/usr/include/utmp.h	Data about who currently is using the system

SEE ALSO

acct(8), acctcms(8), acctcon(8), acctmerg(8), acctprc(8), acctsh(8), runacct(8)

acctcom(1), ed(1) in the *UNICOS User Commands Reference Manual*, Cray Research publication SR–2011

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

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

NAME

gated – Performs routing protocols

SYNOPSIS

gated [-c] [-C] [-n] [-N] [-t*trace_options*] [-f *config_file*] [*trace_file*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The gated routing daemon can be configured to manage one or more routing protocols, including the routing information protocol (RIP) and open shortest path first (OSPF) protocol. The gated daemon replaces routed(8).

Normally, gated detaches from the terminal and runs in the background. However, if trace options are specified without specifying a trace file, gated assumes that tracing is desired to be sent to stderr and remains in the foreground.

The gated daemon accepts the following options and arguments:

-c Specifies that the configuration file will be parsed for syntax errors and then gated will exit. If no errors occurred, gated leaves a dump file in /usr/tmp/gated_dump. The gated daemon does not need to be run as the super user to use the -c option, but it may not be possible to read the kernel forwarding table and interface configuration if it is not run as super user. The -c option implies -tgeneral. All *trace_option* clauses in the configuration file are ignored.

-C Specifies that the configuration file will only be parsed for syntax errors. The gated daemon exits with a status of 1 if there were any errors and 0 if there were not. gated does not need to be run as the super user to use the -C option, but it may not be possible to read the kernel forwarding table and interface configuration if not run as super user.

-f *config_file*
Specifies an alternate configuration file. By default, gated uses /etc/gated.conf.

-n Specifies that gated will not modify the kernel forwarding table. Used for testing gated configurations with actual routing data.

-N Specifies that gated will not daemonize.

-t*trace_options*
Specifies a comma-separated list of trace options to be enabled on startup. If no options are specified, *general* is assumed. No space is allowed between this option and its arguments. This option must be used to trace events that take place before the configuration file is parsed, for example, determining the interface configuration and reading routes from the kernel. See the gated-config(5) man page for valid trace options and a more detailed explanation of tracing.

trace_file
Specifies the file that receives tracing information.

Signal Processing

The gated daemon catches the following signals and does the following special processing:

SIGHUP Reads configuration again. This signal causes gated to reread the configuration file. gated first performs a cleanup of all allocated policy structures. Then the configuration file is parsed again.

OSPF is not capable of reconfiguring; it is shut down and restarted during a reconfiguration. This may have an adverse impact on the routing system.

You can enable and disable any protocol without restarting gated.

SIGINT Takes snapshot of current state. The current state of all gated tasks, timers, protocols and tables are written to /usr/tmp/gated_dump.

On the UNICOS system, this is done by creating (using fork(2)) a subprocess to dump the table information so as not to impact the routing functions of gated.

SIGTERM Graceful shutdown. On receipt of this signal, gated tries a graceful shutdown. All tasks and protocols are requested to shut down.

All protocol routes are removed from the kernel forwarding table on receipt of a SIGTERM signal. Interface routes, routes with RTF_STATIC set (from the route(8) command), and static routes specifying retain remain. To terminate gated with all routes intact, use SIGKILL.

SIGUSR1 Toggle tracing. On receipt of a SIGUSR1 signal, gated closes the trace file. A subsequent SIGUSR1 signal causes it to be reopened, allowing the file to be moved regularly.

It is not possible to use SIGUSR1 if a trace file has not been specified, or tracing is being performed to the stderr file.

SIGUSR2 Check for interface changes. On receipt of a SIGUSR2 signal, gated rescans the kernel interface list looking for changes.

FILES

Many of the default file names in the following list contain the string %s, which is replaced by the name with which gated is invoked. Normally this is gated, but if invoked as gated-test, gated will by default look for /etc/gated-test.conf. These paths may all be changed at compilation time.

These are the default filenames:

/usr/tmp/gated_dump The file to which gated writes status information. The default is /usr/tmp/%s_dump.

/etc/gated.conf The gated configuration file. The default is /etc/%s.conf.

`/etc/gated.pid` The file to which `gated` writes its process ID (PID). The default is `/etc/%s.pid`.

SEE ALSO

`arp`(8), `gdc`(8), `ifconfig`(8), `ospf_monitor`(8), `ripquery`(8), `route`(8)

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

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

COPYRIGHT INFORMATION

NAME

gdc – Operational user interface for gated(8)

SYNOPSIS

gdc [-q] [-t *seconds*] *command*

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The gdc command provides a user-oriented interface for the operation of the gated(8) routing daemon. It provides support for starting and stopping the daemon, for the delivery of signals to manipulate the daemon when it is operating, for the maintenance and syntax checking of configuration files, and for the production and removal of state dumps and core dumps. gdc can reliably determine the running state of gated(8) and produces a reliable exit status when errors occur, making it advantageous for use in shell scripts that manipulate gated(8). Commands executed using gdc and, optionally, error messages produced by the execution of those commands, are logged using the same syslogd(8) facility which gated(8) itself uses, providing an audit trail of operations performed on the daemon.

If installed as a setuid root program, gdc allows nonroot users who are members of a trusted group (by default the gdmaint group) to manipulate the routing daemon while denying access to others. The name of the user and each executed command are logged using the syslogd(8) daemon.

The gdc daemon accepts the following options and arguments:

-q Runs quietly. With this option, informational messages normally printed to standard output are suppressed and error messages are logged using syslogd(8) instead of being printed to the standard error output. This is often convenient when running gdc from a shell script.

-t *seconds* Specifies the time in seconds gdc waits for gated(8) to complete certain operations, in particular at termination and startup. The default value is 10 seconds.

command Specifies the operation that gated(8) performs.

The *command* argument can have one of the following values, all of which cause signals to be delivered to gated(8):

COREDUMP Sends an abort signal to gated(8), causing it to terminate with a core dump.

dump Signals gated(8) to dump its current state into the file /usr/tmp/gated_dump.

interface Signals gated(8) to recheck the interface configuration. gated(8) normally does this periodically, but the facility can be used to force the daemon to check interface status immediately when changes are known to have occurred.

KILL Causes gated(8) to terminate ungracefully. Used most frequently when the daemon has hung.

`reconfig` Signals `gated`(8) to reread its configuration file, reconfiguring its current state as appropriate.

`term` Signals `gated`(8) to terminate after shutting down all operating routing protocols gracefully. Executing this command a second time causes `gated`(8) to terminate even if some protocols have not yet fully shut down.

`toggletrace` If `gated`(8) is currently tracing to a file, causes tracing to be suspended and the trace file to be closed. If `gated`(8) tracing is suspended, causes the trace file to be reopened and tracing initiated. This is useful for moving trace files.

By default `gated`(8) obtains its configuration from a file usually named `/etc/gated.config`. The `gdc` program also maintains these other versions of the configuration file:

`/etc/gated.conf+` The *new* configuration file. When `gdc` is requested to install a new configuration file, this file is renamed `/etc/gated.conf`.

`/etc/gated.conf-` The *old* configuration file. When `gdc` is requested to install a new configuration file, the previous `/etc/gated.conf` is renamed to this name.

`/etc/gated.conf--` The *really old* configuration file. `gdc` retains the previous *old* configuration file under this name.

The following values for the *command* argument perform operations related to configuration files:

`checkconf` Checks `/etc/gated.conf` for syntax errors. This command is usually executed after making changes to the configuration file but before sending a `reconfig` signal to the currently running `gated`(8), to ensure that there are no errors in the configuration which would cause the running `gated`(8) to terminate on reconfiguration. When this command is used, `gdc` issues an informational message indicating whether parse errors occurred or not. If errors did occur, `gdc` saves the error output in a file for inspection.

`checknew` Performs the same operation as `checkconf`, except that the *new* configuration file, `/etc/gated.conf+`, is checked.

`newconf` Moves the `/etc/gated.conf+` file into place as `/etc/gated.conf`, retaining the older versions of the file as described previously. If the *new* configuration file does not exist or otherwise appears impaired, `gdc` does nothing when this command is specified.

`backout` Rotates the configuration files in the newer direction, in effect moving the old configuration file to `/etc/gated.conf`. This command does not execute if the `/etc/gated.conf-` file does not exist or is zero length, or if the operation deletes an existing, nonzero length `/etc/gated.conf+` file.

`BACKOUT` Performs a `backout` operation even if `/etc/gated.conf+` exists and is of nonzero length.

`modeconf` Sets all configuration files to mode 664, owner to `root`, and group to `gdmaint`. This allows a trusted nonroot user to modify the configuration files.

`createconf` If the `/etc/gated.conf+` file does not exist, this command creates a zero length file with the file mode set to 664, owner to `root`, and group to `gdmaint`. This allows a trusted non-root user to install a new configuration file.

The following values for the *command* argument provide support for starting and stopping `gated`(8), and for determining its running state:

`running` Determines if `gated`(8) is currently running by checking if `gated`(8) has a lock on the file containing its process ID (PID), if the PID in the file is within a reasonable range of PIDs, and if a running process has that PID. Exits with zero status if `gated`(8) is running, nonzero otherwise.

`start` Starts `gated`(8). The command returns an error if `gated`(8) is already running. Otherwise it executes the `gated`(8) binary and waits for the delay interval (10 seconds by default, as set with the `-t` option otherwise) or less, until the newly started process obtains a lock on the PID file. A nonzero exit status is returned if an error is detected while executing the binary, or if a lock is not obtained on the PID file within the specified wait time.

`stop` Stops `gated`(8), gracefully if possible, ungracefully if not. The command returns an error (with nonzero exit status) if `gated` is not currently running. Otherwise it sends a `TERM` signal to `gated`(8) and waits for the delay interval (10 seconds by default, or as specified with the `-t` option otherwise) or less for the process to exit. Should `gated`(8) fail to exit within the delay interval, a second `TERM` signal is sent. Should `gated`(8) fail to exit by the end of the second delay interval, a `KILL` signal is sent. This should force immediate termination. The command terminates with a zero exit status when it detects that `gated`(8) has terminated, nonzero or otherwise.

`restart` If `gated`(8) is running, terminates `gated`(8) using the same procedure as for the `stop` command. If `gated`(8) was not running prior to command exeution, or when the previous `gated`(8) terminates, starts a new `gated`(8) process using the procedures described for the `start` command. A nonzero exit status is returned if any step in this procedure fails.

The following values for the *command* argument allow the removal of files created by the execution of some of the previous commands:

`rmcore` Removes any existing `gated`(8) core dump file.

`rmdump` Removes any existing `gated`(8) state dump file.

`rmparse` Removes the parse error file generated when a `checkconf` or `checknew` command is executed and syntax errors are encountered in the configuration file being checked.

BUGS

Many commands work only when `gated`(8) is installed in the system directory in which it was configured.

FILES

`/etc/gated`	The `gated`(8) binary
`/etc/gated.conf`	Current `gated`(8) configuration file
`/etc/gated.conf+`	Newer configuration file
`/etc/gated.conf-`	Older configuration file
`/etc/gated.conf--`	Much older configuration file
`/etc/gated.pid`	Location of PID for `gated`(8)
`/usr/tmp/gated_dump`	State dump file for `gated`(8)
`/usr/tmp/gated_parse`	Location of parse errors for the configuration file
`/usr/tmp`	Location of `gated`(8) core file

SEE ALSO

`gated`(8), `ospf_monitor`(8), `ripquery`(8), `route`(8), `syslogd`(8)

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

COPYRIGHT INFORMATION

NAME

getconfig – Searches the accounting configuration file for the specified argument

SYNOPSIS

/usr/lib/acct/getconfig *label*

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The getconfig command searches the accounting configuration file for *label* and returns the value associated with it. This command is generally used in shell scripts.

By default, getconfig searches the /etc/config/acct_config file. If the shell variable ACCTCONFIG is set to an alternative file, getconfig will search a different configuration file.

EXAMPLES

Example 1: The following example extracts the value for HOLIDAY_FILE from the default configuration file. The shell variable HOLIDAY is set to the extracted value.

```
HOLIDAY=‘getconfig HOLIDAY_FILE‘
```

Example 2: The following example shows how to search an alternative configuration file for the HOLIDAY_FILE variable:

```
HOLIDAY=‘ACCTCONFIG=/tmp/myconfig getconfig HOLIDAY_FILE‘
```

FILES

/etc/config/acct_config Accounting configuration file

SEE ALSO

UNICOS Resource Administration, Cray Research publication SG–2302

NAME

getpal – Gets the privilege assignment list (PAL) category entries of a file

SYNOPSIS

getpal [-c *catlist*] [-p *privlist*] [-t *privtext*] *files...*

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The getpal command displays the privilege assignment list (PAL) category entries of the specified regular file(s).

The output consists of a list of PAL entries, one per line, in the following format:

category_name:*privlist*:*privtext*

category_name is the name of the category (for example, secadm). *privlist* is a character string that specifies a list of zero or more privileges (with multiple privileges separated by commas). A *privlist* value of PRIV_NULL indicates no privileges. A *privlist* value of PRIV_ALL indicates all privileges. *privtext* is a sequence of 0 to 8 characters that represent privilege text. A *privtext* value of TEXT_NULL indicates null privilege text.

If multiple file names are specified for *files*, the following line precedes the PAL information for each file.

#*filename*:

The getpal command accepts the following options and operands:

-c *catlist* Outputs PAL entries for each of the categories specified in *catlist*. If this option is not specified, it outputs PAL entries for any category, including other. *catlist* is a character string that represents one or more category names (for example, secadm). Multiple category names must be separated by commas, with no intervening white space.

-p *privlist* Outputs PAL entries that contain any of the privileges specified in *privlist*. If this option is not specified, or if the PRIV_ALL privilege name is specified, it outputs PAL entries containing any privilege sets, including PRIV_NULL. *privlist* is a character string that represents one or more privilege names (for example, PRIV_MAC_READ). Multiple privilege names must be separated by commas, with no intervening white space.

-t *privtext* Outputs PAL entries that contain the privilege text character sequence specified by *privtext*. If this option is not specified, it outputs PAL entries containing any privilege text value, including TEXT_NULL. *privtext* is a sequence of one to eight alphanumeric characters, or the word TEXT_NULL, that represents privilege text.

files Represents the name(s) of the file(s) whose PALs will be displayed.

If no options are specified, then output is produced for every PAL entry of the specified files.

NOTES

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

Active Category	Action
`system`, `secadm`	Allowed to specify any file.
`sysadm`	Allowed to specify any file, subject to security label restrictions. Shell redirected output is subject to security label restrictions.

If the `PRIV_SU` configuration option is enabled, the super user is allowed to specify any file.

EXIT STATUS

The `getpal` command exits with one of the following values:

Value	Description
0	The requested PAL entries were successfully displayed.
1	A badly formed option or option that is not valid was supplied.
2	When multiple files are supplied, both failure and success occurred.
4	The PAL(s) for the specified file(s) could not be displayed.

EXAMPLES

The following examples assume that `testfile` has the following PAL assigned to it:

```
system:PRIV_ALL:TEXT_NULL
secadm:PRIV_MAC_READ,PRIV_MAC_WRITE:TEXT_NULL
sysadm:PRIV_MAC_WRITE,PRIV_KILL:admin
other:TEXT_NULL
```

Example 1: The following example outputs all PAL entries for the `secadm` category for `testfile` using the `-c` option:

```
$ getpal -c secadm testfile
secadm:PRIV_MAC_READ,PRIV_MAC_WRITE:TEXT_NULL
```

Example 2: The following example displays all PAL entries with the `admin` privilege text for `testfile` using the `-t` option:

```
$ getpal -t admin testfile
sysadm:PRIV_MAC_WRITE,PRIV_KILL:admin
```

Example 3: The following example displays all PAL entries with the PRIV_MAC_WRITE privilege for testfile using the -p option:

```
$ getpal -p PRIV_MAC_WRITE testfile
system:PRIV_ALL:TEXT_NULL
secadm:PRIV_MAC_READ,PRIV_MAC_WRITE:TEXT_NULL
sysadm:PRIV_MAC_READ,PRIV_KILL:admin
```

Example 4: The following example shows getpal executed with no options specified, which displays every PAL entry for testfile:

```
$ getpal testfile
system:PRIV_ALL:TEXT_NULL
secadm:PRIV_MAC_READ,PRIV_MAC_WRITE:TEXT_NULL
sysadm:PRIV_MAC_READ,PRIV_KILL:admin
other:PRIV_NULL:TEXT_NULL
```

Example 5: The following example displays all PAL entries with both the PRIV_MAC_WRITE privilege and the admin privilege text for testfile using the -p and -t options:

```
$ getpal -p PRIV_MAC_WRITE -t admin testfile
sysadm:PRIV_MAC_WRITE,PRIV_KILL:admin
```

Example 6: The following example shows the display when no options are specified and multiple files are specified:

```
$ getpal testfile emptypalfile
# testfile:
system:PRIV_ALL:TEXT_NULL
secadm:PRIV_MAC_READ,PRIV_MAC_WRITE:TEXT_NULL
sysadm:PRIV_MAC_WRITE,PRIV_KILL:admin
other:PRIV_NULL:TEXT_NULL
# emptypalfile:
other:PRIV_NULL:TEXT_NULL
```

SEE ALSO

setpal(8)

NAME

getprivs – Gets the privilege sets of a file

SYNOPSIS

getprivs [-a] [-f] [-s] *files...*

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The getprivs command displays the allowed, forced, and set-effective privilege sets of the specified regular file(s).

Output consists of a list of privilege sets, one per line, in the following format:

priv_set_id:*privlist*

priv_set_id can be a, f, or s indicating the allowed, forced, and set-effective privilege sets, respectively. *privlist* is a character string that specifies a list of privileges. Multiple privilege names are separated by commas with no intervening white space. When no privileges are set, the value of *privlist* is the PRIV_NULL character string. The PRIV_ALL character string represents the list of all privileges.

If multiple file names are specified for *files*, the following line precedes the PAL information for each file:

#*filename*:

The getprivs command accepts the following options and operand:

-a Displays the allowed privilege set of each specified file.

-f Displays the forced privilege set of each specified file.

-s Displays the set-effective privilege set of each specified file.

files Represents the name(s) of the file(s) whose privilege set(s) will be displayed.

If no options are specified, then all privilege sets are displayed for each specified file.

NOTES

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

Active Category	Action
system, secadm	Allowed to specify any file.
sysadm	Allowed to specify any file, subject to security label restrictions. Shell redirected output is subject to security label restrictions.

If the PRIV_SU configuration option is enabled, the super user is allowed to specify any file.

EXIT STATUS

The getprivs command exits with one of the following values:

Value	Description
0	The specified privilege state was successfully reported.
1	A badly formed option or option that is not valid was supplied.
2	When multiple files are supplied, both failure and success occurred.
4	Privilege state for the specified file(s) could not be obtained.

EXAMPLES

The following examples assume that *testfile* has the following privilege states:

```
ALLOWED: PRIV_NULL
FORCED: PRIV_ALL
SET-EFFECTIVE: PRIV_MAC_READ,PRIV_MAC_WRITE
```

Example 1: The following example shows getprivs executed with no options specified, which results in all the privilege sets being displayed for testfile:

```
$ getprivs testfile
a:PRIV_NULL
f:PRIV_ALL
s:PRIV_MAC_READ,PRIV_MAC_WRITE
```

Example 2: The following example displays the forced and set-effective privilege sets of testfile using the -f and -s options:

```
$ getprivs -f -s testfile
f:PRIV_ALL
s:PRIV_MAC_READ,PRIV_MAC_WRITE
```

Example 3: The following example displays the set-effective privilege set of testfile using the -s option:

```
$ getprivs -s testfile
s:PRIV_MAC_READ,PRIV_MAC_WRITE
```

Example 4: The following example displays the allowed and forced privilege sets of testfile using the -f and -a options:

```
$  -f -a testfile
a:PRIV_NULL
f:PRIV_ALL
```

Example 5: The following example shows the display when no options are specified and multiple files are specified:

```
$ getprivs testfile noprivsfile
# testfile:
a:PRIV_NULL
f:PRIV_ALL
s:PRIV_MAC_READ,PRIV_MAC_WRITE
# noprivsfile:
a:PRIV_NULL
f:PRIV_NULL
s:PRIV_NULL
```

SEE ALSO

setprivs(8)

NAME

gettable – Gets NIC format host tables from a host

SYNOPSIS

/etc/gettable [-v] *host* [*outfile*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The gettable command obtains the Network Information Center (NIC) standard host tables from a *nicname* server (a name server that is running on a machine that is a NIC).

The gettable commands supports the following option and operands:

-v Gets only the version number instead of the complete host table and puts the output complete host table in the file *outfile* or, by default, hosts.ver.

host The indicated *host* is queried for the table.

outfile If retrieved, the table is placed in the file *outfile* or, by default, hosts.txt.

The gettable command operates by opening a TCP/IP connection to the port that is indicated in the service specification for the nickname server. A request is then made for all names, and the resulting information is placed in the output file.

The gettable command is best used in conjunction with the htable(8) command, which converts the NIC standard file format to that used by the network library look-up routines.

SEE ALSO

htable(8), named(8)

NAME

getty – Sets up an interactive connection

SYNOPSIS

```
/etc/getty [-m] [-L minlvl [-maxlvl]] [-C mincmp [-maxcmp]] [-t timeout] line [speed]
/etc/getty -c file
```

IMPLEMENTATION

Cray PVP systems

DESCRIPTION

The getty command is invoked by init(8). It is the second process in the series (*init-getty-login-shell*) that ultimately connects a user with the UNICOS operating system. Initially, getty prints the login message field for the entry it is using from /etc/gettydefs. getty reads the user's login name and invokes the login(1) command with the user's name as the argument.

On a secure system, the following options specify the security label range at which the line may be used:

-m
: Indicates that the line should be marked as a multilevel device. If the -m option is omitted, the line is labeled single-level.

-L *minlvl* [-*maxlvl*]
: Specifies the security level range at which the line will be labeled. The *minlvl* and *maxlvl* values are specified as decimals. If *maxlvl* is omitted, the maximum level is set equal to the minimum. If the -L option is omitted, the minimum and the maximum level are set equal to the system minimum level.

-C *mincmp* [-*maxcmp*]
: Specifies the compartment range at which the line will be labeled. The *mincmp* and *maxcmp* values are specified as integers that correspond to the binary bitmap of compartments. Octal or hexadecimal values can be given by prefixing the value with 0 or 0x, respectively. If *maxcmp* is omitted, the maximum compartment value is set equal to the minimum. If the -C option is omitted, the minimum and maximum compartment values are set to 0.

The getty command accepts the following options and operands:

-t *timeout*
: Specifies that getty should exit if the open on the line succeeds and no one types anything in *timeout* number of seconds.

line
: The name of a tty line in /dev to which getty is to attach itself. getty uses this string as the name of a file in the /dev directory to open for reading and writing.

speed
: Optional; a label to a speed and tty definition in the /etc/gettydefs file. This definition tells getty what the login message should look like and provides the initial tty settings. The default *speed* is 300.

`-c` *file* A check option; scans the *file* as if it were scanning `/etc/gettydefs` and prints out the results to the standard output. If there are any unrecognized modes or improperly constructed entries, it reports them. If the entries are correct, it prints the values of the various flags.

Finally, `login` is called with the user's name as an argument. Additional arguments may be typed after the login name. These are passed to `login`, which places them in the environment (see `login`(1)).

NOTES

The `getty` command is run on all terminals (ttys) directly attached to the IOS.

FILES

`/etc/gettydefs`

SEE ALSO

`init`(8)

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

`gettydefs`(5), `inittab`(5), `tty`(4) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

NAME

hddmon – HIPPI disk device monitor and control program

SYNOPSIS

hddmon [-o] [-s *slot*]

IMPLEMENTATION

Cray PVP systems

DESCRIPTION

The hddmon program is used to monitor and control HIPPI disk array devices. The hddmon program can be used to monitor statistics detailing hdd(4) device usage and to control the read/write mode and the up/down state of the device.

The hddmon program has two modes of operation: interactive and single-pass. Running hddmon without the -o option brings up an interactive menu-driven refreshed display. Specifying the -o option does a single pass and writes statistics to stdout.

-o The -o option prints the main view screen. If the -s option is also indicated, detailed statistical information about the device occupying the specified slot is printed.

-s *slot* Specifying the -s option by itself selects the indicated *slot* number as the current slot of the display.

The mnu Package

In interactive mode, the hddmon program makes use of a menu-based display package called mnu(4). The mnu(4) package provides command input and screen refresh capabilities using the following primitives:

Primitive	Description
<TAB> or right arrow	menu right
<BACK SPACE> or left arrow	menu left
<RETURN>	next menu level or execute menu item
<ESC>	back to first menu level
<CONTROL-F> or <PAGE DOWN>	next display page
<CONTROL-B> or <PAGE UP>	previous display page
<CONTROL-D> or down arrow	display down 1 line
<CONTROL-U> or up arrow	display up 1 line
?	help

The first letter of a given menu item selects and executes that menu item. A help facility is available for nearly all menu items; to invoke the help facility, select the desired menu item and type `?`. For information on the the `mnu`(4) package, see the `mnu`(4) man page.

The Main View Screen

When you invoke `hddmon` in interactive mode, the main view screen appears. The main view screen lists all HIPPI disk array facilities arranged in the order in which they were opened. Each HIPPI disk facility that has been previously initialized is assigned a "slot" number. The current slot number is denoted by a `*` and is used to view detailed statistics or direct control to a particular device. A sample main view screen appears as follows:

```
hddmon:                                                          09/14/94 13:24:32
slot   view   display   help   configure   els   quit
                                                                 Page: 0  Line: 0
+slot   -slot   #slot
                                                    Sectors moved      Errors
slot iopth fcty rprt ifield type  state     mode     reads   writes   rec  unr
---- ----- ---- ---- ------ ----  --------- ----  -------- --------  ---- ----
   0  0130 0x10 0x10      6 HD64  open   up   rw       8961    19252     0    0
*  1  0130 0x10 0x11      6 HD64  open   up   rw       4096    34194     0    0
   2  0130 0x10 0x00     21 HD32  clsd   up   rw          1        0     0    0
```

The `iopth`, `fcty`, `rprt`, `ifield`, and `type` designate the I/O path, facility number, raid partition, ifield, and device type, respectively.

The `state` of the device indicates whether the device is open or closed and whether the run state is up, down, or suspend. Normally the state appears open and up.

The `mode` display indicates the read/write mode of the device. Possible modes are `rw` (read/write), `ro` (read only), or `na` (no allocate). The default mode is read/write. The mode is used by the filesystem to indicate the read and write capabilities of the device.

The last four columns summarize device activity and indicate the number of sectors transferred as well as recovered and unrecovered errors.

Main Menu Items

The main `hddmon` menu includes the following options:

Option	Description
`slot`	Selects the current slot.
`view`	Selects from several screens of statistical information about the device in the current slot.
`display`	Controls display primitives such as the refresh rate and page number in case the display information overflows the available window.
`help`	Displays menu primitives.

`configure` Sets the state and mode for the current device. Changing the device state can come in handy when the disk array or I/O subsystem is hung up and needs to be restarted. This procedure is detailed in the subsection "Array/IOS restart procedure."

`els` (CRAY J90 systems) Displays statistics and error information and configures the logical channels up and down. The menu screen that this item invokes is described in the subsection "The `els` display."

`quit` Exits the `hddmon` program.

Nearly all of the menu items have help text associated with them. For that reason, not all menu possibilities are listed here. Do not be afraid to explore. Select the menu item and type `?` to enter help mode. When in help mode, you can move easily from one menu selection to another.

Array/IOS Restart Procedure

If it becomes necessary to restart the HIPPI disk array and/or the IOS, you may be able to do this without a system interruption or, possibly, without losing a request, by using the following procedure.

1. Set the proper current slot and set the device state to suspend by selecting the `configure -> state -> suspend` menu item.
2. Restart the array and/or restart the IOS on the operator workstation (OWS):
   a. Use `eboot`(8) to reboot `eiop.hpi`.
   b. Use `econ`(8) to configure the IOP down.
   c. Use `econ`(8) to configure the IOP up.
3. Put the device in `up` state by selecting the `configure -> state -> up` menu item.

See the `econ`(8) and `eboot`(8) man pages for further information on using these commands on the OWS.

Configuration

The `hddmon` command opens the control node for `hdd` devices. By definition the control node is `/dev/ddd/hdd`. It is made implicitly by the `mkspice`(8) command by default whenever `mkspice`(8) is executed or explicitly by the `mkspice`(8) command as follows:

```
mkspice -t HD64 hdd
```

The `/dev/ddd/hdd` control node is assigned a minor number of 0. That means that you should avoid configuring your `/dev/hdd` nodes with a minor number of zero. Failure to do so could cause `hddmon` to exit with an `EBUSY` (device busy) error.

The `els` Display

The following screen appears when you select the `els` menu item and the `display` menu item on the `els` screen:

```
                                                                  04/27/95 14:07:30
main   display   select   configure
                                                                  Page: 0  Line: 0

                                        Errors                 I/O Transfers (bytes)
index  chan    path state  rejects  conn  retries  fail    read  Swrite  Cwrite
----- ------- ---- ----- -------- ----- -------- ----- ------- ------- -------
* 0   064/067 disk open         42     0       42     0    3.1M    7.8M     0.0
             tape close         0     0        0     0    0.0     0.0      0.0
  1   104/107 disk open         30     0       30     0    2.8M    6.1M     0.0
             tape close         0     0        0     0    0.0     0.0      0.0
```

The `els` menu includes the following options:

Option	Description
`main`	Returns you to the top-level `hddmon` display.
`display`	Displays the information as shown in the example.
`select`	Increments, decrements, or selects a specific channel for configuration (configuring up or down). The channel that is currently selected is indicated by a `*`.
`configure`	Configures the logical channel used for IPI-3 disk I/O up or down.

Each channel in the `els` display corresponds to a memory-HIPPI channel on the CRAY J90 series. Each memory-HIPPI can support both IPI-3 disk and tape traffic.

The `els` display includes the following categories:

Category	Description
index	An index number that identifies a specific memory-HIPPI channel. The memory-HIPPI channel that is currently selected is identified by a `*` in the leftmost column. In the example display above, the memory-HIPPI channel with the index of 0 is currently selected. The `configure` option uses the currently selected channel to perform channel up and down functions.
`chan`	The physical input and output memory-HIPPI channel of the form: input/output (for example, 064/067) The channel numbers are in octal.
`path`	Identifies the driver using the channel. Only disk and tape are supported.
`state`	The current state of the disk or tape driver using the channel. The possible values for this field are:

Value	Description
`open`	The channel path has been opened for use.
`close`	The channel path is closed.

DOWN The channel path has been configured down.

`rejects` The number of connection rejects reported by the memory-HIPPI driver.

`conn` The number of not connected errors reported by the memory-HIPPI driver.

`retries` The number of retries attempted by the CRAY J90 series IPI-3 pseudo driver.

`fail` The number of errors reported for the channel path.

`read` The number of read bytes for this channel path.

`Swrite` The number of simple write (single-packet) bytes for this channel path.

`Cwrite` The number of complex write (double-packet) bytes for this channel path.

Invoking the `select` option from the `els` menu adds the following options to the display:

Option	Description
`+channel`	Increments the index to the next channel.
`-channel`	Decrements the index to the previous channel.
`#channel`	Selects a specific channel using the index number.

Invoking the `configure` option from the `els` menus adds the following options to the display.

Option	Description
`up`	Configures the disk channel path up.
`down`	Configures the disk channel path down.

The `configure` option configures only the disk channel path. The tape channel path must be configured by using `hpi3_config`(8).

When the channel path is configured down, it must be configured back up before the channel path is available for normal use.

FILES

`/usr/src/uts/cmd/disk/hddmon.c`

`/usr/src/uts/cmd/disk/mnu.c`

`/usr/include/sys/mnu.h`

`/usr/include/sys/hddstat.h`

SEE ALSO

`mkspice`(8)

`eboot`(8), `econ`(8) in the *Support System Reference Manual*, Cray Research publication SR–3077

`hdd`(4), `mnu`(4) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

NAME

hit – Performs a HYPERchannel interface test

SYNOPSIS

/etc/hit [-lrhw] [-1ikpstvx] [-a *addr*] [-d *dsize*] [-n *npass*] [-z *device*] *local remote*

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The hit command performs one of three loopback tests on a Network Systems Corporation (NSC) HYPERchannel network adapter. Messages can be looped back: in the local adapter, in any other adapter on the HYPERchannel network, or in another host computer on the HYPERchannel network. hit can also be used with front-end interface (FEI) and VME devices.

The hit command can operate in one of two modes:

- Active mode: sends a message and waits for it to be echoed back (see the -l, -r, and -h options).
- Passive mode: waits for a message and echos it back (see the -w option).

In active mode, hit performs one or more passes, each of which consists of the following steps:

1. Sends a message proper.
2. Sends an associated data segment (if requested); see the -d option.
3. Receives a message proper.
4. Receives an associated data segment (if requested); see the -d option.
5. Checks data in the message proper (if requested); see the -p option.
6. Checks data in the associated data segment (if requested); see the -d and -p options.
7. Updates the data pattern (if requested); see the -i option.

In passive mode, hit performs one or more passes, each of which consists of the following steps:

1. Receives a message proper.
2. Receives an associated data segment (if requested); see the -d option.
3. Sends the message proper.
4. Sends the associated data segment (if requested); see the -d option.

Some of the command line options can alter these sequences in active and passive modes.

The `hit` command accepts the following options (one of them must be specified):

`-l` Performs loopback to the *local* adapter, using ‘‘transmit local message’’ functions. Only the *local* adapter address is required.

A130 note: The A130 is not capable of performing local loopback. The IOS model E driver for the A130 does not do the loopback; therefore, this test will fail.

`-r` Performs loopback to a *remote* adapter, using transmit message functions with byte 8 of the message proper (numbering from byte 0) set to 0xFF. Both *local* and *remote* adapter addresses are required.

`-h` Specifies the active side of a loopback test to a remote `host`, using transmit message functions. It is the responsibility of the remote host to turn the messages around. Both *local* and *remote* adapter addresses are required.

Note: The remote host must be running `hit -w` (or its equivalent) for this option to work correctly.

`-w` Specifies the passive, or `waiting`, side of a loopback test to a remote host. Only the *local* adapter address is required.

Note: The network driver assigns a logical channel number that `hit` must incorporate in the local adapter address. The `hit` command issues a message indicating the actual local adapter address it is using. You must specify the correct address when starting the active partner (that is, `hit -h`. See the About NSC Addresses subsection for more information.

You can use the remaining options to modify default settings.

`-1` Specifies one-way data. This option is used with two cooperating `hit` processes (one with the `-h` option, the other with the `-w` option). The sending process sends a message proper plus data, but expects only a message proper in reply. The receiving process reads in a message proper plus data, but sends only a message proper in reply. This simulates a common occurrence during file transfers, where data is moving in only one direction. Do not confuse this option with the `-k` option (described later).

`-i` Increments each byte of the data pattern at the end of each successful pass. See the Data Formats subsection for a description of the data pattern. This slows down the rate of message exchange. The default is to used a fixed data pattern. You cannot select this option if you have selected the `-w` option.

`-k` Runs `hit` in a very fast and unrestrained manner. Use this option with two cooperating `hit` processes (one with the `-h` option, the other with the `-w` option). (Do not confuse this with the `-l` option.) The sending process sends a message proper (and perhaps data) but never tries to read a reply. The receiving process reads in a message proper (and perhaps data) but never tries to send a reply. This option obtains maximum transfer rates.

Note: Not all drivers handle this kind of maximum-rate I/O. This option was developed to test the Cray Research UNICOS driver. Other drivers may drop messages (or worse).

`-p` Checks each byte of the data pattern at the end of each pass. `hit` compares the received data with the transmitted data. If any discrepancies exist, the number of bytes in error is reported. If you select verbose mode (see the `-v` option), each byte in error is printed, along with the expected value. This slows down the rate of message exchange. The default is not to check received data. You cannot select this option if you have already selected the `-w` option.

`-s` Stops the test after an error is detected. The default is to terminate the current pass of the test when an error is detected and proceed to the next pass.

`-t` Prints timing information after the last pass has completed. The elapsed time is printed (to six decimal places, but the precision depends on the granularity of the system clock). A data transfer rate, in bytes per second and bits per second, is also printed *if* associated data segments were sent and received (see the `-d` option), *and* no errors were detected. The transfer rate calculation is: (size of associated data segment in bytes) × (number of passes) × 2 ÷ (elapsed time in seconds).

`-v` Writes verbose output to `stdout`, slowing down the rate of message exchange. The default is to print only error messages.

`-x` Writes debug output to `stdout`, slowing down the rate of message exchange. This option is useful only if you have a `hit` source listing. The default is to suppress debugging information.

`-a` *addr* Allows you to specify the Internet network number. *addr* is a decimal class A Internet network number from 1 through 127; the default is 84. Usually, the default value is the only one needed; therefore, this option may be safely ignored. See the About Internet Addresses subsection for more information.

`-d` *dsize* Allows you to specify, as a decimal integer *dsize*, the associated data size in bytes. It ranges from 0 (no data) to 16,384 bytes. (This maximum data size is a compile-time parameter and can be changed.) The default is 4096 bytes. For local and remote adapter loopback, the data size is limited by the adapter memory size, typically 4096 bytes. For remote host loopback, the data size is limited by the loopback software running on the remote host.

`-n` *npass* Allows you to specify the number of passes as a decimal integer *npass*. The default is 100 passes.

`-z` *device* Allows the character special device name to be specified. The default is `/dev/hy00` for logical channel 0, `/dev/hy01` for logical channel 1, and so on. If you select this option, the named device must agree with the *local* operand. The default device name may also be modified at compile time.

You must specify one or both of the adapter addresses. See the About NSC Addresses subsection for more information.

local Local adapter address, specified as 4 hexadecimal digits. For example, A400 adapter unit 0x13, port 1, is specified as `1301`.

remote Remote adapter address, specified as 4 hexadecimal digits. For example, A130 adapter unit 0x40, logical channel 4, is specified as `4004`.

About NSC Addresses

NSC HYPERchannel addresses consist of 16 bits. The high-order 8 bits are always the adapter unit number, which is configured with thumbwheel switches on the back of the adapter. The low-order 8 bits vary, depending on adapter type. For example, the A400 adapter uses the low-order 2 bits for a port number (up to four host computers can be connected to a single A400). The remaining 6 bits are used for software routing (the logical channel number).

You need to know the local adapter unit number and port number. The network driver fills in the software routing field in the address. For example, if you are using adapter unit 0x13 port 1, you would specify address 1301. The driver may assign you logical channel number 1; in this case, your actual address would be 1305. On Cray PVP systems, you cannot use logical channel number 0 (for example, `/dev/comm/. . ./lp00` or `/dev/hy00`.) It is reserved by the IOS for USCP; if you use this path, `hit` will report an error 5.

You also need to know the remote adapter unit number. For remote adapter loopback (see the `-r` option), you do not need to know what kind of adapter it is. You can ignore the low-order 8 bits of the address. For example, remote adapter unit 0 x 40 would be specified as 4000. For remote host loopback (see the `-h` option), you need to know both the remote adapter type and the software on the remote host. They dictate the value in the low-order 8 bits of the remote adapter address. If `hit` is running on the remote host in passive mode (see the `-w` option), it tells you what the full address is.

About Internet Addresses

Each HYPERchannel network is assigned a class A Internet network number. (This is a restriction that will be lifted in the future.) `hit` must specify this number to the `hy`(4) network driver. `hit` is configured with a single network number; the default is `84`. Some sites may have more than one HYPERchannel network. In this case, each network has its own class A number (for example, the Cray Research development network is 84, and the production network is 86). Therefore, `hit` accesses the development network by default, and requires the option `-a 86` to access the production network.

Data Formats

The message proper always contains a 48-byte data area, which begins in byte 16 (numbering from byte 0). It is always filled with the default (fixed) data pattern. If you specify an associated data segment (with the `-d` option), the data segment is also filled with the data pattern. The data pattern consists of one-byte integers 0, 1, 2, 3 ..... Message proper byte 16 contains 0, byte 17 contains 1, and so on. Similarly, data segment byte 0 contains 0, byte 1 contains 1, and so on. If more than 256 bytes are in the data segment, the pattern repeats every 256 bytes.

If you specify the `-i` option, each byte in the pattern is incremented after each pass. For the second pass, message proper byte 16 contains 1, byte 17 contains 2, and so on.

MESSAGES

`-l or -r option: data size limit=`*number*`.`
`-w option: cannot specify -i or -p.`
`Associated data rate=`*number* `bytes/second.`
(*number* bits/second). (Only if `-t` option selected)

`Bind error, errno=`*number*`, lchan=`*number*`.`	(Network driver only)
`Can't establish network connection.`	(Network driver only)
`Can't open` *device*`.`	(Not if using network driver)
`Elapsed time=`*number* `seconds.`	(Only if `-t` option selected)
`Invalid data size specified.`	(Only if `-d` option selected)
`Invalid local address` *number*`.`	
`Invalid network specified.`	(Only if `-a` option selected)
`Invalid number of passes.`	
`Invalid remote address` *number*`.`	
`Local adapter address is` *hex_number*`.`	
`Local address not specified.`	
`No logical channels available.`	(Network driver only)
`One of -lrhw must be specified.`	
`Received message from` *hex_number*`.`	(Only if `-w` option selected)
`Remote address not specified.`	
`Setsockopt error, errno=`*number*`.`	(Network driver only)
`Socket error, errno=`*number*`.`	(Network driver only)
`Still waiting for connection.`	(Only if `-w` selected)
`Stop, error detected.`	(Only if `-s` option selected)
`Unknown option` *c*`.`	
`Usage: hit ...`	

All other errors are logged with the time of day, last function, pass number, and a cumulative error count. The last function is the last system call (`sendto` (see `send`(2)), `recvfrom` (see `recv`(2)), `write`(2), or `read`(2)).

As a rule, the network driver does not report errors to the user, but rather writes messages to the console. Look at the system console and `/usr/adm/messages` if there are problems.

Theory

Presently, `hit` can operate with one of two device drivers:

- The network driver (4.2 BSD default)
- The Cray Research UNICOS driver (System V default)

Network Driver Operation

`hit` opens a raw socket. Each active pass consists of a `sendto` (see `send`(2)) call to write out the message proper (and associated data, if any), followed by a `recvfrom` (see `recv`(2)) call to read in the message proper (and associated data, if any). Because few errors are reported to the user level, a 30-second time is started before each network operation. If the timer expires, you should examine the system console and `/usr/adm/messages` to see whether the driver has reported any problems.

Cray UNICOS Driver Operation

The `hit` command opens an entry in the `/dev` directory. The name is `/dev/hy00` for logical channel 0, `/dev/hy01` for logical channel 1, and so on. The logical channel number is taken from the two low-order hexadecimal digits of the local adapter address. Each active pass consists of a `write`(2) call to write out the message proper (and associated data, if any), followed by a `read`(2) call to read in the message proper (and associated data, if any).

NOTES

The test may be stopped at any time with the `SIGINT` signal, which is usually <`CONTROL-c`>.

If `hit` refuses to run, try master-clearing the local adapter manually (press the red reset button on the back of the adapter).

You should match the options on two cooperating copies of `hit`. If a passive copy is started for 1000 passes, the active copy should also specify 1000 passes. The passive copy should always be started first.

EXAMPLES

A good way to use `hit` is to try local loopback first. This checks out the driver, interface, and local adapter. If this works, try remote adapter loopback. This ensures the two adapters can talk. Try remote loopback with several destination adapters, from each host on the network. Finally, try remote host loopback. This proves that two hosts can talk to each other.

Example 1: The following example performs a local loopback with 4-Kbyte data segments for 100 passes:

```
hit -lt 1301
```

Timing information is reported when the test is complete. The local adapter is A400 unit 0x13 port 1.

Example 2: The following example performs a remote loopback with 4-Kbyte data segments for 100 passes:

```
hit -rt 1301 2100
```

Timing information is reported when the test is complete. The local adapter is A400 unit 0x13 port 1, and the remote adapter is A130 unit 0x21.

Example 3: The following example performs a remote loopback with no data for 10,000 passes:

```
hit -rips -d0 -n10000 1301 C100
```

The data pattern in the message proper is checked and then incremented at the end of each pass. The test halts if any error is detected. The local adapter is unit A400 0x13 port 1, and the remote adaptor is A130 unit 0xC1.

Example 4: The following example starts the passive side of a remote host loopback, with 4-Kbyte data segments, for 100 passes:

```
hit -wt 1301 2301
```

Timing information is reported when the test is complete. The local adapter is A400 unit 0x13 port 1, and the remote adapter is unit A130 0x23 logical channel 1. The `hit` command issues a message showing the actual local adapter address (including the logical channel number), as in the following example:

```
hit: local adapter address is 1305
```

Exxample 5: The following example starts the active side of a remote host loopback, with 4-Kbyte data segments, for 100 passes (the remote adapter address was taken from the `hit` diagnostic message in the preceding example):

```
hit -ht 2301 1305
```

Timing information is reported when the test is complete. The local adapter is A130 unit 0x23, logical channel 1, and the remote adapter is A400 unit 0x13, logical channel 1, port 1.

FILES

`/dev/hy`*nn*	Cray Research UNICOS HYPERchannel driver entry (*nn* = logical channel number)

SEE ALSO

`hy`(4), `vme`(4) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

NAME

hpi3_clear – Clears a IPI-3/HIPPI packet driver device

SYNOPSIS

/usr/lib/hpi3_clear *devicename*

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The hpi3_clear command terminates all outstanding requests to *devicename*. Requests that have finished processing but have not been returned to the user are discarded. No further user requests to the device are processed until the device is closed. When using the hpi3_clear command, you must specify a *devicename*.

EXIT STATUS

The hpi3_clear command completes successfully, 0 is returned; otherwise, a nonzero value is returned. Where possible, this exit status code is normalized to the last three digits. Exit status values are documented in the *Tape Subsystem User's Guide*, Cray Research publication SG–2051.

FILES

/dev/hpi3/*devicename*	IPI-3/HIPPI interface devices
/dev/hpi3/reqt	IPI-3/HIPPI interface devices
/etc/config/hpi3_config	IPI-3/HIPPI configuration file

SEE ALSO

hpi3_config(8), hpi3_option(8), hpi3_start(8), hpi3_stat(8), hpi3_stop(8)

hpi3(4) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

Tape Subsystem Administration, Cray Research publication SG–2307

Tape Subsystem User's Guide, Cray Research publication SG–2051

NAME

hpi3_config – Configures a IPI-3/HIPPI channel up or down

SYNOPSIS

Cray PVP systems with IOS model E:

/usr/lib/hpi3_config -c *channel-pair* -C *cluster* -i *iop state*

CRAY EL series and CRAY J90 series:

/usr/lib/hpi3_config -c *channel-pair state*

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The hpi3_config command configures a IPI-3/HIPPI channel up or down. The operands are as follows:

-c *channel-pair*	Specifies the channel pair that will be configured.
-C *cluster*	Specifies the cluster in which the channel is configured.
-i *iop*	Specifies the IOP in which the channel is configured.
state	Specifies HPI-3 channel state (either up or down).

EXIT STATUS

The hpi3_config command completes successfully, 0 is returned; otherwise, a nonzero value is returned. Where possible, this exit status code is normalized to the last three digits. Exit status values are documented in the *Tape Subsystem User's Guide*, Cray Research publication SG–2051.

FILES

/dev/hpi3/*devicename*	IPI-3/HIPPI interface devices
/dev/hpi3/reqt	IPI-3/HIPPI interface devices
/etc/config/hpi3_config	IPI-3/HIPPI configuration file

SEE ALSO

hpi3_clear(8), hpi3_option(8), hpi3_start(8), hpi3_stat(8), hpi3_stop(8)

hpi3(4) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

Tape Subsystem Administration, Cray Research publication SG–2307

Tape Subsystem User's Guide, Cray Research publication SG–2051

NAME

hpi3_option – Modifies a IPI-3/HIPPI packet driver option(s)

SYNOPSIS

/usr/lib/hpi3_option [-a *maximum-number-async-responses*] [-c *maximum-number-cmdlst*] [-i *maximum-number-iop-processes*] [-r *maximum-number-non-cmdlst*] [-t on|off]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The hpi3_option command modifies one or more IPI-3/HIPPI packet driver options. The options are as follows:

-a *maximum-number-async-responses*
: Specifies the maximum number of asynchronous responses that may be outstanding per device. Users cannot enable asynchronous responses that exceed this value.

-c *maximum-number-cmdlst*
: Specifies the maximum number of command list requests that may be outstanding per device. If the number of command list requests issued exceeds this value, an error is returned.

-i *maximum-number-iop-processes*
: Specifies the maximum number of processes that can open an I/O processor (IOP) device concurrently. If this limit is exceeded, an error is returned.

-r *maximum-number-non-cmdlst*
: Modifies the maximum number of noncommand list requests that may be outstanding per device.

-t on|off
: Specifies the tracing state (either on or off).

NOTES

If a process has an IOP device open, you cannot modify the maximum number of processes that may open an IOP.

EXIT STATUS

The hpi3_option command completes successfully, 0 is returned; otherwise, a nonzero value is returned. Where possible, this exit status code is normalized to the last three digits. Exit status values are documented in the *Tape Subsystem User's Guide*, Cray Research publication SG–2051.

FILES

`/dev/hpi3/`*devicename*	IPI-3/HIPPI interface devices
`/dev/hpi3/reqt`	IPI-3/HIPPI interface devices
`/etc/config/hpi3_config`	IPI-3/HIPPI configuration file

SEE ALSO

`hpi3_clear`(8), `hpi3_config`(8), `hpi3_start`(8), `hpi3_stat`(8), `hpi3_stop`(8)

`hpi3`(4) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

Tape Subsystem Administration, Cray Research publication SG–2307

Tape Subsystem User's Guide, Cray Research publication SG–2051

NAME

`hpi3_start` – Starts the IPI-3/HIPPI packet driver subsystem

SYNOPSIS

`/usr/lib/hpi3_start` [`-f` *config-file*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `hpi3_start` command informs the IPI-3/HIPPI packet driver and, for Cray PVP systems with IOS model E (IOS-E), the IPI-3/HIPPI IOP driver of the physical organization of each IOP. For each IOP configured, a corresponding IOP file is created. This file is used to issue packets that affect the IOP as a whole. For each device attached to the IOP, `hpi3_start` creates a corresponding device file. Device files are used to issue packets that affect the device.

The user provides the configuration in a file, which describes the IOP devices, channels, slaves, devices, and options.

The `hpi3_start` command accepts the following option:

`-f` *config-file* Specifies the configuration file. The default configuration file is `/etc/config/hpi3_config`.

For Cray PVP systems with IOS-E, the default configuration file is of the following format:

```
-IOPS
IOP-name   cluster   IOP

-CHANNELS
IOP-name  input-channel  output-channel  state input-channel-timeout
  output-channel-timeout connection-timeout

-SLAVES
slave-name  IOP-name input-ch1:output-ch1[,input-ch2:output-ch2] i-field

-DEVICES
device-name   slave-name  low-facility-addr  high-facility-addr

-OPTIONS
option    option-value
```

For the CRAY EL series and CRAY J90 series, the default configuration file is of the following format:

```
-CHANNELS
input-channel  output-channel  state  input-channel-timeout
output-channel-timeout  connection-timeout

-SLAVES
slave-name  input-ch1:output-ch1[,input-ch2:output-ch2]  i-field

-DEVICES
device-name  slave-name  low-facility-addr  high-facility-addr

-OPTIONS
option  option-value
```

NOTES

You must stop all configured IOP drivers before executing the `hpi3_start` command.

EXIT STATUS

The `hpi3_start` command completes successfully, 0 is returned; otherwise, a nonzero value is returned. Where possible, this exit status code is normalized to the last three digits. Exit status values are documented in the *Tape Subsystem User's Guide*, Cray Research publication SG–2051.

FILES

`/dev/hpi3/`*device-name*	IPI-3/HIPPI interface devices
`/dev/hpi3/reqt`	IPI-3/HIPPI interface devices
`/etc/config/hpi3_config`	IPI-3/HIPPI configuration file

SEE ALSO

`hpi3_clear`(8), `hpi3_config`(8), `hpi3_option`(8), `hpi3_stat`(8), `hpi3_stop`(8)

`hpi3`(4) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

Tape Subsystem Administration, Cray Research publication SG–2307

Tape Subsystem User's Guide, Cray Research publication SG–2051

NAME

`hpi3_stat` – Displays device statistics

SYNOPSIS

Cray PVP systems with IOS model E:

`/usr/lib/hpi3_stat` [`-c`] [`-C` *cluster*] [`-d` *devname*] [`-i` *iop*] [`-t`]

CRAY EL series and CRAY J90 series:

`/usr/lib/hpi3_stat` [`-c`] [`-d` *devname*] [`-t`]

All Cray Research systems:

`/usr/lib/hpi3_stat` [`-o`]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `hpi3_stat` command displays device status, configuration, packet driver table, or the packet driver options statistics. The status, configuration, or table values for all devices can be displayed, or a partial display can be requested (for example, you can request a cluster, I/O processor (IOP), or a particular device to be displayed). The status of all devices is the default display.

The `hpi3_stat` command accepts the following options:

`-c` Specifies that the configuration will be displayed. By default, the status is displayed. The `-c`, `-o`, and `-t` options are mutually exclusive.

`-C` *cluster* Specifies the cluster in which all configured devices should be displayed.

`-d` *devname* Specifies the device for which information is displayed.

`-i` *iop* Specifies the IOP in which all configured devices should be displayed.

`-t` Specifies that the packet driver table be displayed. By default, the status is displayed. The `-c`, `-o`, and `-t` options are mutually exclusive.

`-o` Specifies that the options should be displayed. You cannot specify the `-o` option with any other options.

When you request device configuration (`-c`) information, the following information is displayed:

device The device name

ioc The cluster in which the device is configured

iop The IOP in which the device is configured

chpair A channel pair in which the device is configured in the following format:

input-channel:output-channel

ichst The status of the input channel:

- `down` Interrupts logically disabled
- `wconn` Waiting for connection
- `wpkt` Waiting for an IPI-3 packet
- `read` Reading data
- `dump` Reading and ignoring data

ochst The status of the output channel:

- `down` Interrupts logically disabled
- `up` Interrupts enabled
- `conn` Connection established
- `disc` Connection terminated
- `cmd` Read of command complete
- `wcmd` Write of command complete
- `wdta` Write of data complete

sl The ordinal of the slave in which the device is attached

lfc The low facilities address

hfc The high facilities address

If you do not specify `-c`, `-t`, or `-o`, the following information is displayed:

device The device name

ioc The cluster in which the device is configured

iop The IOP in which the device is configured

chpair A channel pair in which the device is configured in the following format:

input-channel:output-channel

ichst The status of the input channel:

- `down` Interrupts logically disabled
- `wconn` Waiting for connection
- `wpkt` Waiting for an IPI-3 packet
- `read` Reading data
- `dump` Reading and ignoring data

ochst The status of the output channel:

`down`	Interrupts logically disabled
`up`	Interrupts enabled
`conn`	Connection established
`disc`	Connection terminated
`cmd`	Read of command complete
`wcmd`	Write of command complete
`wdta`	Write of data complete

If you specify `-t`, the following table information is displayed:

device The device name

ord The table ordinal

flag The flags used by the IPI-3/HIPPI packet driver:

0001	The user has open the device file
0002	An IOP request timed out
0004	Packet interface was enabled
0010	The device is open (IOP open)
0020	A signal was registered
0100	A clear device is in progress
0200	The device has been cleared
0400	Device processing is waiting for an interrupt

sig The signal to send to the user when a packet is returned from the IOP

pid The ID of the process that has the device open

lock The number of locks on the process

rsyn The resynchronization code

async The number of queued asynchronous packets (that is, asynchronous packets returned by the IOP but not received by the user)

enabl The number of asynchronous packets that are currently enabled

usr The number of outstanding user requests

ios The number of outstanding IOP requests

cmd The number of command list requests outstanding to the IOP

EXIT STATUS

The `hpi3_stat` command completes successfully, 0 is returned; otherwise, a nonzero value is returned. Where possible, this exit status code is normalized to the last three digits. Exit status values are documented in the *Tape Subsystem User's Guide*, Cray Research publication SG–2051.

FILES

`/dev/hpi3/`*devicename*	IPI-3/HIPPI interface devices
`/dev/hpi3/reqt`	IPI-3/HIPPI interface devices
`/etc/config/hpi3_config`	IPI-3/HIPPI configuration file

SEE ALSO

`hpi3_clear`(8), `hpi3_config`(8), `hpi3_option`(8), `hpi3_start`(8), `hpi3_stop`(8)

`hpi3`(4) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

Tape Subsystem Administration, Cray Research publication SG–2307

Tape Subsystem User's Guide, Cray Research publication SG–2051

NAME

hpi3_stop – Stops the IPI-3/HIPPI subsystem

SYNOPSIS

/usr/lib/hpi3_stop [-u] [-w]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The hpi3_stop command shuts down the IPI-3/HIPPI packet driver, disables further opens of the IOPs and IPI-3 devices, waits for current activity to stop, and if requested, stops the IOP drivers.

The hpi3_stop command accepts the following options:

-u Leaves the IOP drivers up.

-w Waits for all activity to stop and for all devices to be closed. By default, if the IPI-3/HIPPI packet driver is not idle, an error is returned.

EXIT STATUS

The hpi3_stop command completes successfully, 0 is returned; otherwise, a nonzero value is returned. Where possible, this exit status code is normalized to the last three digits. Exit status values are documented in the *Tape Subsystem User's Guide*, Cray Research publication SG–2051.

FILES

/dev/hpi3/*devicename* IPI-3/HIPPI interface devices

SEE ALSO

hpi3_clear(8), hpi3_config(8), hpi3_option(8), hpi3_start(8), hpi3_stat(8)

hpi3(4) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

Tape Subsystem Administration, Cray Research publication SG–2307

Tape Subsystem User's Guide, Cray Research publication SG–2051

NAME

hpmall – Reports hardware performance statistics for entire machine workload

SYNOPSIS

/etc/hpmall [-d] [-r] [-t *second*] [-V]

/etc/hpmall [-g *group*]

IMPLEMENTATION

Cray PVP systems

DESCRIPTION

The hpmall command reports the Hardware Performance Monitor (HPM) statistics that have been accumulated for all user processes on the machine. Additionally, for super users, it can set the default HPM group under which all processes will be run. hpmall writes its output to standard output.

The hpmall command accepts the following options:

-d Displays HPM statistics for each CPU on a multi-CPU machine. The default is to combine the statistics for all CPUs before they are reported.

-g *group* Number of the HPM monitor group under which all processes will be run at the next connection to the CPU (not used on the CRAY C90 or CRAY T90 series). Note that this option can be executed only by a super user. If this option is specified, all other options are ignored.

group can be any of the following values:

0 Execution summary

1 Hold issue conditions

2 Memory activity

3 Vector events and instruction summary

-r Generates "raw" output suitable for postprocessing by other tools, such as awk(1). See the *Guide to Parallel Vector Applications*, Cray Research publication SG–2182, for a description of this format. If this option is not specified, the output is written as a report.

-t *second* The amount of time to delay between making two samples of the HPM counters. If this option is not specified, the HPM statistics reported are for the entire machine since boot-up. If this option is specified, hpmall samples the counters, waits for *second* seconds, and then samples again. It then reports the difference between these two samples. If the value given for *second* exceeds 1800 (30 minutes), the delay value is set to 1800.

-V Displays the current version of hpmall, as well as a short copyright notice.

NOTES

The meanings of the HPM statistics and their implications are discussed in detail in the *Guide to Parallel Vector Applications*, Cray Research publication SG–2182. On Cray PVP systems (except CRAY C90 and CRAY T90 series), users can override the default HPM counter group for their programs by executing the `hpm`(1) command or by using the perftrace library. Therefore, if you set the default group by using `hpmall`, a user can still accumulate statistics under a different group. These statistics will appear in `hpmall` reports, regardless of the default HPM group that you have set.

On Cray PVP systems (except CRAY C90 and CRAY T90 series), note that if the default HPM counter group is changed to be other than 1, no wait semaphore time will be recorded for accounting, because counter group 1 is used to measure the wait semaphore time.

EXAMPLES

Example 1: The following example shows how to generate a report to the file `hpm.all` that shows the combined HPM statistics for all CPUs on the current machine, since boot-up.

```
/etc/hpmall > hpm.all
```

Example 2: The following example shows how to generate a report to the file `hpm.all` that shows the combined HPM statistics for all CPUs on the current machine, for a 30-second period.

```
/etc/hpmall -t 30 > hpm.all
```

Example 3: The following example shows how to generate a report to the file `hpm.all` that shows the HPM statistics for each CPU on the current machine, since boot-up.

```
/etc/hpmall -d > hpm.all
```

Example 4: The following example shows how to generate raw data for the whole machine workload and post-process it with `perfview`.

```
/etc/hpmall -r > hpm.raw
perfview hpm.raw
```

Example 5: The following example shows how to change the HPM counter group under which all subsequent processes will run. This can be executed only by a super user. (No report is generated.)

```
/etc/hpmall -g 0
```

Note: The previous example is not used on the CRAY C90 and CRAY T90 series.

FILES

`/dev/hpm_all`

SEE ALSO

hpm(4) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

Guide to Parallel Vector Applications, Cray Research publication SG–2182

NAME

hpmflop – Reports hardware performance statistics gathered for user processes

SYNOPSIS

/etc/hpmflop [-a] [-d *date*] [-t] [-u *maxusers*] [-V] [*datafile*]

/etc/hpmflop [-c] [-d *date*] [-m] [-p] [-t] [-u *maxusers*] [-V] [*datafile*]

IMPLEMENTATION

Cray PVP systems

DESCRIPTION

The hpmflop command reports the average megaflops achieved by user processes, based on statistics gathered by special instrumentation from the Hardware Performance Monitor (HPM). These statistics are optionally gathered on a site-wide basis for all normally terminated user programs on the machine. hpmflop writes its report output to standard output.

Review of the output of this command allows the site to determine which users and which of their programs might be using large amounts of CPU time with low Megaflop rates. Thus, these users can be contacted and encouraged to optimize their programs.

The hpmflop command accepts the following options and operands:

-a Reports on all processes for all user IDs. By default, hpm(1) reports processes belonging only to the user invoking hpmflop.

-c Reports a summary sorted by user ID and by CPU seconds used.

-d *date* Reports on processes for a particular day for the user invoking hpmflop. Used with the -a option, this reports on all processors for that day. Default: reports all processes chronologically.

date must be of the form "*Mmm dd*":

- The value must be surrounded by double quotes because of the embedded blank.
- *Mmm* is the standard alphabetic abbreviation for a particular month as defined by ctime(3C)
- The single space between month and day is required.
- *dd* must be two digits. Include a leading zero for days 1 through 9.

```
hpmflop -d"Jul 08" rawfile
```

-m Reports a summary by user ID sorted by MFLOPS.

-p Reports a summary by user ID and process sorted by process; it can be used with -c and -m.

`-t` Reports on today's processing; it can be used with `-c` and `-m` for daily summaries for the invoking user or with `-a` for a daily list of all processes. It reports processes belonging only to the user invoking `hpmflop` chronologically.

`-u` *maxusers*
Allocates an internal table large enough to hold *maxusers* different user IDs. The default is 1,000 different user IDs. If the raw data file contains records for more than this number of different user IDs, `hpmflop` will issue a fatal error message and terminate. *maxusers* must be a positive, nonzero, decimal value.

`-V` Displays the current version of `hpmflop`, as well as a short copyright notice.

datafile Name of the file containing HPM information for `hpmflop` to process. The information was written to a site-selected file by the `hpmdump` global data-gathering feature described in the following section. By default, the name of the data file is `/usr/spool/hpm.data`.

Data Gathering

The data gathering itself is controlled by the contents of the following files:

```
/lib/segdirs/def_ld
/lib/segdirs/def_seg
```

To enable loading of the global gathering code, these files must include the following `SEGLDR` directive:

```
hardref=_hpmdumpg
```

To activate the global gathering, an additional step is required. You must create a file to receive the data. The file name used is a fixed value: `/usr/spool/hpm.data` and must have its privileges set so that every terminating process has access to and can write to it. This file will increase in size during production processing, so the site personnel with responsibilities for maintaining the file should also be responsible for removing older versions.

This file must exist (even if empty) before the data-gathering code can write statistics into it. If your site chooses to use a different file for this data gathering, the file `/usr/spool/hpm.data` must be linked to your site's chosen file, using the `ln`(1) command.

The algorithm used by the embedded data gathering code may cause some process terminations to be missed and not written into the data file. However, the missed processes will not be statistically significant.

Note: The following two paragraphs do not apply to the CRAY C90 or the CRAY T90 series.

Before the data gathering scheme can supply useful information, the default HPM counter group for all processes should be set to group 0. You can set the default HPM counter group using the `hpmall`(8) command. Note that such a setting is active only until the next system restart. It is acceptable to set the default HPM counter group to group 3, but the megaflops values reported by `hpmflop` will be for only vector floating-point operations.

The site default HPM counter group can also be set permanently during UNICOS installation.

To prevent large numbers of trivial processes from appearing in the collected data, a default minimum total CPU time of 5 seconds is set in the data-gathering software. This value can be overridden by setting an environment variable as follows. The *cputime* value must be a decimal number of seconds. Only processes whose total accumulated CPU time exceed this threshold will be considered for inclusion in the common file.

`/etc/profile`:

```
HPM_MT=cputime ; export HPM_MT
```

`/etc/cshrc`:

```
setenv HPM_MT cputime
```

Raw-format Output

The `hpmflop` command processes data generated by the global data-gathering feature, which is written to the common statistics file. Each record of output contains the data from one HPM counter group for a terminating process. If a process ran under more than one counter group, it may have more than one record written.

The records are written in ASCII with a fixed number of fields in each record. The fields are separated by white space. The fields described below are decimal numbers unless otherwise indicated.

Sample record for the CRAY C90 and CRAY T90 series:

```
hpmg 701379651 13612 834 ./a.out 92905263 11927052 74532647 1109
            180091749 13797955 0 0 3061091 121009 54064351 201810
            31801275 658 8474328 1506763 603712 626749 7333326 13238
            19590 16380 755 12761 11190 225 665 120000043 33 30000000
            120002196 60022765
```

Field	Description
1	HPM global record marker (the string `hpmg`).
2	Time stamp taken when the process terminated. This number can be passed to `ctime`(3C) to see a printable date/time value.
3	Process identification number (PID) of the process whose execution is responsible for the HPM statistics.
4	User identification number (UID) of the user who is running this process. Additional information, such as the user's name, can be extracted from the running system using the `getpwuid` (see `getpwent`(3C)) call.
5	Name of the executable file that was run to generate these HPM statistics. This is a string.
6-37	HPM counter values, with field 6 being counter 0, and field 37 being counter 31.

Sample record for Cray PVP systems (except CRAY C90 and CRAY T90 series):

```
hpmg 701379576 19728 834 ./a.out 1 315 1 7295375 3099449 85613587
            58800482 175050 553262 176035219
```

Field	Description
1	HPM global record marker (the string `hpmg`).
2	Time stamp taken when the process terminated. This number can be passed to `ctime`(3C) to see a printable date/time value.
3	Process identification number (PID) of the process whose execution is responsible for the HPM statistics.
4	User identification number (UID) of the user who is running this process. Additional information, such as the user name, can be extracted from the running system, using the `getpwuid` (see `getpwent`(3C)) call.
5	Name of the executable file that was run to generate these HPM statistics. This is a string.
6	HPM counter group. This will be a number from 0 through 3.
7-14	HPM counter values, with field 7 being counter 0 of the group, and field 14 being counter 7 of the group.
15	Total CPU time accumulated for this process, when run under the HPM counter group given in field 6. Processes may accumulate time under other HPM groups, and such times would be reported in separate termination records.

Note that the raw-format output contains more statistics than the megaflop rate of the processes. This raw data can be processed by `awk`(1) scripts, such as those generated by the `perfscripts`(1) command.

NOTES

`hpmflop` must be executed on the same type of machine, with the same clock speed, on which the original data was gathered. Do not mix machine types. If the user ID values are mapped differently on different machines, likewise, the results from this command may not be useful.

The meanings of the HPM statistics and their implications are discussed in detail in the *Guide to Parallel Vector Applications*, Cray Research publication SG–2182.

Troubleshooting

The data-gathering instrumentation is activated by loading of user programs with the `hardref` SEGLDR directive. However, several errors can occur when this code is preparing the data for writing to the common file. To conserve user program memory space, this error handling is quite primitive. If your site has set up all of the required configuration and data files but no data appears on the common file, some error analysis will be required.

The data-gathering instrumentation code stores error flags in global memory locations in every executable program in which it has been loaded. The easiest way to determine the current processing error is to run a trivial program under a debugger and check the value of the global variable `_hpm_global_error`. In some cases, a system `errno` also occurs, and this value will be stored in the global variable `_hpm_global_sys_errno`.

The possible decimal values for `_hpm_global_error` are as follows:

1 The program could not open the `/dev/hpm_mult` device. Either the machine does not have this device defined, or else some greater I/O error occurred. The `_hpm_global_sys_errno` variable will be set to the `errno` associated with the failed open.

2 The program could not read the `/dev/hpm_mult` device correctly. This may be due to some mismatch in the expected data size, or some other I/O error. The `_hpm_global_sys_errno` variable will be set to the `errno` associated with the failed read.

4 This user execution did not accumulate enough CPU time to meet the criteria for minimum CPU time. The minimum is the decimal environment variable `HPM_MT` or 5 seconds.

5 Unable to open the global data file (for writing) whose name is `/usr/spool/hpm.data`. The `_hpm_global_sys_errno` variable contains the system `errno` value for the failed attempts to open.

The global data file is site-maintained, and must have world-write permissions.

Failures to perform the writes on the global data file are not noted by the termination code. This type of failure can be checked independently by site personnel.

If the variable for `_hpm_global_error` is not present in the program being tested, the feature was not loaded correctly. Check the `/lib/segdirs/def_ld` and `/lib/segdirs/def_seg` files carefully.

Security Considerations

If your site chooses to enable the `hpm` global data gathering feature, the following security issues should be considered:

Because the global data file is world-writeable, the opportunity exists for a user to abuse it, including: Altering the data or rendering it unusable, and/or filling up the file system in which the data is written, making the file system unusable.

For sites relying on the mandatory access control (MAC) protection mechanism, this feature should not be enabled, because it provides a direct channel to bypass the MAC policy enforced by system.

EXAMPLES

The following command requests statistics for all processes on the machine for January 17. Assume that a data file was accumulated for all processes on the machine for dates that included January 17th. The data is contained in the file `raw.data`.

```
hpmflop -d"Jan 17" -a raw.data
```

Output would consist of one or more report lines in the following form:

```
Jan 17 15:45 rds       124.26 Mflops      65.4 Mvops 0.12s ./a.out
```

The fields in the report lines have the following meanings:

Field	Meaning
1-3	Date and time of the normal process termination.
4	User ID as generated by the function `getpwuid` (see `getpwent`(3C)). If this raw performance data is being processed on a different machine from that on which it was gathered, this field may be shown as (NULL). Such a problem could be caused by the mismatch of user ID numbers on the different machines.
5, 6	Number of average megaflops achieved during the execution of the process.
7, 8	Number of average megaflops achieved for vector arithmetic operations.
9	Number of total CPU seconds executed by the process.
10	Command used to invoke the process.

Review of such a report allows the site to determine which users and which of their programs might be using large amounts of CPU time with low megaflop rates. Thus, these users' programs can be targeted for further optimizations.

SEE ALSO

`hpmall`(8) for instructions for setting the default counter group

`awk`(1), `perfscripts`(1) for information about commands that can process raw data produced by the global data-gathering feature *UNICOS User Commands Reference Manual*, Cray Research publication SR–2011

`ctime`(3C) for information about date and time formatting
`getpwent`(3C) for information about user IDs and user names in the *UNICOS System Libraries Reference Manual*, Cray Research publication SR–2080

Guide to Parallel Vector Applications, Cray Research publication SG–2182, for descriptions of all the performance tools

NAME

hsxconfig – Configures HIPPI and HSX channel interfaces

SYNOPSIS

/etc/hsxconfig *device_name status*

IMPLEMENTATION

Cray PVP systems (except CRAY J90 series and CRAY EL series)

DESCRIPTION

The hsxconfig command lets an operator configure HSX high-speed external communication channels up and down. This is useful in preventing simultaneous use of HSX channels and online tape devices, which can result in performance degradation. It also configures the ANSI High Performance Parallel Interface (HIPPI). See hippi(4) for more information.

Setting a channel up or down is appropriate only on the logical path zero device (/dev/hsx0/i00 or /dev/hsx0/o00). The inode permission bits determine who may perform this task.

Additionally, hsxconfig can configure a logical path in auto header mode. In auto header mode, the HSX driver adds a header to each output block and removes it from each input block, allowing user processes to share a hardware channel transparently.

Auto header mode is appropriate only on nonzero logical paths. Only a super user can change the header mode.

EXAMPLES

Example 1: The following example configures the HSX channel referred to as /dev/hsx0/i00 up:

```
hsxconfig /dev/hsx0/i00 up
```

Example 2: The following example configures the HSX channel referred to as /dev/hsx0/i00 down:

```
hsxconfig /dev/hsx0/i00 down
```

Example 3: The following example sets auto header mode on the HSX logical path referred to as /dev/hsx0/i01:

```
cd /dev/hsx0
hsxconfig i01 auto
```

Example 4: The following example restores header processing to the user on the HSX logical path referred to as /dev/hsx0/i01:

```
cd /dev/hsx0
hsxconfig hsx0/i01 hdr
```

SEE ALSO

`hippi`(4), `hsx`(4) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

NAME

`htable` – Converts NIC standard format host tables

SYNOPSIS

`/etc/htable` [`-c` *connected_nets*] [`-l` *local_nets*] [*files*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `htable` command converts host *files* in the format specified in Internet RFC 810 to the format used by the network library routines. The `hosts`, `networks`, and `gateways` files are created as a result of running `htable`. The `gethostent`(3C) routines use the `hosts` file in mapping host names to addresses. The `getnetent`(3C) routines use the `networks` file in mapping network names to numbers.

If any of the files `localhosts`, `localnetworks`, or `localgateways` are present in the current directory, the file's contents is prepended to the output file without interpretation. Of these, only the `gateways` file is interpreted. This allows sites to maintain local aliases and entries that are not usually present in the master database. Only one gateway to each network is placed in the `gateways` file; a gateway listed in the `localgateways` file overrides any in the input file.

The `htable` command accepts the following options:

`-c` *connected_nets*	Specifies a list of networks to which the host is directly connected.
`-l` *local_nets*	Specifies a list of networks to be treated as local networks.
files	Specifies files to be converted.

If you use the `gateways` file, a list of networks to which the host is directly connected is specified with the `-c` option. The networks, separated by commas, may be given by name or in Internet-standard dot notation (for example, `-c arpanet,128.32,local.ether.net`). `htable` includes only gateways that are directly connected to one of the networks specified or that can be reached from another gateway on a connected net.

If the `-l` option is given with a list of networks (in the same format as for `-c`), these networks are treated as local, and information about hosts on local networks is taken only from the `localhosts` file. Entries for local hosts from the main data base are omitted. This allows the `localhosts` file to completely override any entries in the input file.

If you omit *file*, `htable` reads from standard input.

The `htable` command is best used in conjunction with the `gettable`(8) command, which retrieves the Network Information Center (NIC) database from a host.

SEE ALSO

`gettable`(8), `named`(8)

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

NAME

hyroute – Sets the Internet address to hardware address mapping

SYNOPSIS

/etc/hyroute *interface* [-c] [-d] [-l] [-p] [-s] [-D] [*file*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The hyroute command manipulates the host or gateway address to hardware address mapping.

The hyroute command accepts the following operands and options:

interface Specifies the name of the interface to which this command refers.

-c Compares the system's current information to that contained in *file*.

-d Dumps the system's table (used for debugging routing code).

-l Displays on standard error a trace of the parsing of *file*.

-p Prints a digested version of *file*.

-s (Set) Reads *file* and sets the system's database according to the information in the file.

-D Dumps the system's table in a readable format.

file Specifies the input file. This operand is necessary for all options other than -d. If you do not specify a file name, or if the minus sign (-) is encountered as an argument, hyroute reads from standard input.

The input file is free format. Comment lines begin with a * or # symbol in column 1. Statements end with a semicolon. Three valid statement verbs are direct, gateway, and c_format.

EXAMPLES

The HYPERchannel-DX series of adapters can contain an IP routing engine to perform IP routing between the attached Cray Research system and network media such as Ethernet, FDDI, and T1/T3 links. If you are configuring a Cray Research system with one of these types of adapters, you must place an entry in the configuration file to point to the IP router instead of placing an individual entry for each destination host, as is done for hosts directly attached to HYPERchannel media. For these types of adapters, the configuration file must contain only an entry for the Cray Research system itself and an entry for the IP router. The format of this entry is the same as it is for normal HYPERchannel hosts; however, you should obtain the value to place into the *dest* field of the direct statement from the Network Systems Corporation personnel on site. The address can vary, depending on the configuration of the adapter and the IP routing engine that is being used to route IP packets.

The following statement describes a HYPERchannel host that can be reached directly from an adapter:

```
direct host dest control access [mtu];
```

The *host* value can be a host name (see `hosts`(5)). The *host* can be an Internet address specified in dot notation (see `inet`(3C)). The values *dest*, *control*, and *access* are hexadecimal numbers, and the optional *mtu* value is a decimal number. The data is sent to HYPERchannel address *dest*, using a control value of *control* and an access code of *access* (see adapter manuals for details). The *mtu* field is the maximum size HYPERchannel packet that the host can receive.

The specified remote adapter and the local adapter must both be connected to one or more common trunks or connected to trunks that are connected with link adapters. The following statement describes a host that may be reached indirectly through any one of the hosts indicated by *gate*n:

```
gateway host gate1 gate2 gate3 . . . ;
```

The hosts listed are not gateways in the formal sense (they do not run the Internet gateway protocols), but they are hosts on the HYPERchannel that can bridge between subsections of the HYPERchannel network.

The following statement causes `hyroute` to interpret all subsequent numbers in the input file as integer constants expressed in C syntax. For example, a leading 0x signals a hexadecimal number (such as 0xFF); a leading 0 indicates an octal number (such as 0377); otherwise, the number is decimal (such as 255).

```
c_format;
```

The following statement describes a host that can be reached on an FEI-3 or low-speed channel:

```
direct host dest unused unused [mtu];
```

The following statement describes a host that can be reached over a HIPPI or HSX channel:

```
direct host hwaddr readdev writedev [mtu];
```

The low-order 8 bits of *readdev* and *writedev* contain the minor device numbers of the logical paths to open for reading and writing. The `hwaddr` is the logical channel to use when sending to the specified host.

A sample file follows:

```
* comment
direct azure     6100 0 0 4160;
direct bronze    6101 0 0 4160;
direct cyber     2100 1100 0 4160;
direct dadcad    6102 0 0 4160;
direct tekcad    2400 1100 0 4160;
direct tekcrd    2201 1100 0 4160;
direct tekid     2500 1100 0 4160;
direct teklabs   2200 1100 0 4160;
gateway iddic    tekcrd teklabs cyber tekcad tekid;
gateway iddme    tekcrd teklabs cyber tekcad tekid;
gateway metals   tekcrd teklabs cyber;
```

SEE ALSO

ifconfig(8)

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

hippi(4), hosts(5), hsx(4), hy(4), vme(4) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

NAME

`ifconfig` – Configures network interface parameters

SYNOPSIS

`/etc/ifconfig` *interface* [*address_family*] [*address* [*destination*]] [*parameters*]

`/etc/ifconfig` *interface* [*address_family*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `ifconfig` command assigns an address to a network interface and configures network interface parameters. You must use it at boot time to define the network address of each interface present on a machine; use `ifconfig` also to redefine an interface's address or other parameters.

If you use the format shown in the first synopsis, `ifconfig` sets the configuration for the specified interface; if you use the format shown in the second synopsis, `ifconfig` displays the configuration for the specified interface.

The `ifconfig` command accepts the following arguments:

interface Specifies a string of the form *name-unit* (for example, `fddi0`).

address_family Specifies a name that represents a protocol; this argument is necessary when an interface may receive transmissions from differing protocols. Possible value is `inet`. With the first format, if you omit *address_family*, it sets `inet` configuration. With the second format, if you omit *address_family*, it displays `inet` configuration.

address For the DARPA Internet family, this option specifies the *address* as either a host name present in the host name database, `hosts`(5), or a DARPA Internet address expressed in the Internet standard dot notation.

destination Specifies the address of the other end of a point-to-point network.

parameters Sets directives that can further specify network configuration.

You can set the following parameters by using `ifconfig`:

`admin` This option marks an interface as restricted to authorized users only. Creating a socket with `PRIV_ADMIN` effective on PAL-based systems, or UID `root` on `PRIV_SU` systems, enables a socket for communication on this interface. In a future release, an additional `setsockopt` system call will be required to enable communication on the socket. This option also prevents forwarding of IP packets to and from the interface.

`-admin` Disables the `admin` option.

`alias` Establishes an additional network address for this interface. This is sometimes useful when you are changing network numbers and you want to accept packets addressed to the old interface.

`authority` *byte0*`[:`*byte* `...]`
Labels the interface to specify the authorities that are allowed on packets that are accepted or sent over the interface. The authority list can consist of up to 8 bytes, with each byte separated by a colon. The last byte must have the low-order bit set to 0, and all preceding bytes must have the low-order bit set to 1. The values are specified in hexadecimal notation.

`arp` (`ifconfig` accepts this value but it is not relevant to any Cray Research interface.) Enables the use of the address resolution protocol in mapping between network-level addresses and link-level addresses (default).

`-arp` (`ifconfig` accepts this value but it is not relevant to any Cray Research interface.) Disables the use of the Address Resolution Protocol.

`bg` Retries attempt to set the interface in the background if the first attempt fails.

`broadcast` *addr*
(Internet only) Specifies the address to use to represent broadcasts to the network. The default broadcast address is the address with a host part of all 1's.

`compart` *min*`[-`*max*`]`
Labels the interface to allow only packets that have at least *min* compartments, and no more than *max* compartments, to be accepted or sent over the interface. If *max* is omitted, it is assumed to be equal to *min*. You can specify the *min* and *max* values in hexadecimal notation by preceding the number with 0x; you can specify them in octal notation by preceding the number with 0; or you can specify them in decimal. See the NOTES section for operational information.

`debug` Enables driver-dependent debugging code; usually, this turns on extra console error logging.

`-debug` Disables driver-dependent debugging code.

`delete` Removes the network address specified. This would be used for an incorrectly specified alias or for one that is no longer needed.

`down` Marks an interface down. When an interface is marked down, the system does not try to transmit messages through that interface. If possible, the interface is also reset to disable reception. This action does not automatically disable routes that use the interface.

`hwloop` For packets destined for the local interface that support hardware loopback, use hardware loopback instead of software loopback.

`iftype` *type* (Cray Research systems only) Sets the interface type to one of the following:

`hy` NSC HYPERchannel (see `hy`(4) and `np`(4))

`hippi`	High Performance Parallel Interface (see `hippi`(4))
`hsx`	High-speed external communications (see `hsx`(4))
`n130`	NSC N130 protocol (see `hy`(4) and `np`(4))
`vme`	FEI-3 (see `vme`(4))
pvc	Permanent Virtual Circuit (see `atmarp`(8))
q2931	ATM standard signalling protocol
spans	FORE Systems proprietary signalling protocol

`level` *min*`[-`*max*`]`
Labels the interface to allow only packets that are within the range *min* to *max* to be accepted or sent over the interface. If *max* is omitted, it is assumed to be equal to *min*. The *min* and *max* values are mnemonic level names (see `secnames`(3C)), or they can be specified in decimal. The range of allowed values includes `syslow` and `syshigh`. See the NOTES section for operational information.

`metric` *n* Sets the routing metric of the interface to *n*; the default is 0. The routing protocol uses the routing metric. Higher metrics make a route less favorable; metrics are counted as additional hops to the destination network or host.

`mtu` *size* Changes the size of the read buffers posted to the low-level driver. The *size* argument represents the size available to the IP layer for *datagrams* (that is, data packets), not including link-level or physical frame headers. The size is rounded up to a multiple of the word size and a warning message printed, if necessary.

For GigaRing I/O systems only, the following maximum limits apply for each media type:

Ethernet 1500 bytes (same as default)

FDDI 4500 bytes (default = 4352)

HIPPI 65536 bytes (default = 65280)

ATM 65536 bytes (default = 9180)

If the size of the posted read buffer is too small to receive an entire datagram, the datagram is discarded and the following error message is logged:

```
WARNING: if_fddi.c: fddi0: Bad read: errno = 5
```

This condition can cause applications that use TCP connections (such as `ftp`) to hang, repeatedly retransmitting the data.

The *size* argument must be greater than the write buffer size on all directly connected hosts. Presumably, all hosts on the network use the same write buffer size for a given host. The `ifconfig` command compares the *size* argument to the host's write buffer size in the `hyroute` table. If the host's write buffer size is greater than the *size* argument, the new `mtu` size is accepted, but the following warning message is printed:

```
ifconfig: WARNING - New mtu may be too small.
```

See the `hyroute` command input file `direct` statement for more information on the write buffer size.

This parameter does not apply to the Ethernet and FDDI network interfaces on the CRAY EL series and CRAY J90 series.

`netmask` *mask*
For Internet:

Specifies how much of the address to reserve for subdividing networks into subnetworks. The *mask* includes the network part of the local address and the subnet part, which is taken from the host field of the address. You can specify the *mask* as one hexadecimal number with a leading 0x, with a dot-notation Internet address, or with a pseudo-network name listed in the network table `networks`(5). The mask contains 1's for the bit positions in the 32-bit address that are used for the network and subnet portions, and 0's for the host portion. The mask should contain at least the standard network portion, and the subnet field should be contiguous with the network portion.

`ptp` For GigaRing systems only:

This option flags the interface as a point-to-point interface, causing the HIPPI to be used in "dedicated" mode and all packets to be transmitted in "hold" mode. This mode assumes only TCP is using the HIPPI channel, there is no HIPPI switch in the network, and the network is connected point-to-point to another host.

For all Cray Research systems:

If a destination address is supplied, you can omit this flag, but a warning message is issued. If this flag is set and no destination address is offered, an error message is displayed and `ifconfig` fails.

`-ptp` Turns off the Point-to-point Interface flag.

`rbuf` *bufcnt*
Changes the maximum number of buffers that can be posted to the driver for reads. You can see the current value by using the `netstat -iv` command.

Care should be taken when using this option, as there is no limit set for the *bufcnt* argument. Whatever value the user enters will be used by the system.

This parameter does not apply to the Ethernet and FDDI network interfaces on the CRAY EL series and CRAY J90 series.

`trailers`
(`ifconfig` accepts this value, but it is not relevant to any Cray Research interface.) Requests the use of a trailer link-level encapsulation when sending (default). If a network interface supports trailers, the system, when possible, encapsulates outgoing messages in a manner that minimizes the number of memory-to-memory copy operations that the receiver performs. Currently, only Internet protocols use this parameter.

`-trailers`
(`ifconfig` accepts this value, but it is not relevant to any Cray Research interface.) Disables the use of a trailer link-level encapsulation.

`up` Marks an interface up. This parameter enables an interface after the `ifconfig down` command is issued. The up action occurs automatically when the first address is set on an interface. If the interface is reset after it has been previously marked down, the hardware is reinitialized.

`wbuf` *bufcnt*
Changes the maximum number of buffers that can be posted to the driver for writes. Use the `netstat -iv` command to see the current value.

Care should be taken when using this option, as there is no limit set for the *bufcnt* argument. Whatever value the user enters will be used by the system.

This parameter does not apply to the Ethernet and FDDI network interfaces on the CRAY EL series and CRAY J90 series.

NOTES

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

Active Category	Action
`system`, `secadm`, `sysadm`	Allowed to observe and configure network interface parameters.

If the `PRIV_SU` configuration option is enabled, the super user is allowed to observe and configure network interface parameters.

The `ifconfig` command displays the current configuration for a network interface when no optional parameters are supplied. If an address family is specified, `ifconfig` reports the details specific only to that address family.

The security label of the interface is displayed such that label information that is within the user's security label reange is shown. Only appropriately authorized users can see the actual label information. Only appropriately authorized users can modify the configuration of a network interface.

The security label can be increased only while an interface is configured up. Restricting a label requires the interface to be configured down. When an interface is restricted, routes for that interface are bounded by the new label. If a route is outside of the new label, the route is deleted automatically.

MESSAGES

Messages indicate that the specified interface does not exist, the requested address is unknown to the interface, potential problems exist with the mtu size, or the nonprivileged user tried to alter an interface's configuration.

SEE ALSO

`brc`(8), `hyroute`(8), `initif`(8)

`netstat`(1B), `privtext`(1) in the *UNICOS User Commands Reference Manual*, Cray Research publication SR–2011

`hosts`(5), `intro`(4) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

UNICOS Networking Facilities Administrator's Guide, Cray Research publication SG–2304

NAME

ift – Reads user Factory Flaw table and reports flaws

SYNOPSIS

/etc/ift [-f] [-p] [-s *streams*] *special*

IMPLEMENTATION

Cray PVP systems with I/O subsystem model E

DESCRIPTION

The ift command reads permanent flaw information from a CE cylinder and writes it to standard output as an ASCII Flaw table (aft). The *special* operand is a special file that corresponds to the disk's CE cylinder number 1. By default, ift reads the User Flaw table and reports only nonslippable flaws.

The ift command accepts the following options:

-f Reads the Factory Flaw table rather than the User Flaw table.

-p Reports the physical location of all flaws, even those that were slipped.

-s *streams*
In a case such as an array, when a device is made of individual member drives, each with its own flaw table, *streams* is a bit mask that designates the drives from which flaw tables will be read. Bit 2^0 is drive 0, Bit 2^1 is drive 1, and so on.

NOTES

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

Active Category	Action
system, secadm, sysadm	Allowed to use this command.

If the PRIV_SU configuration option is enabled, the super user is allowed to use this command.

EXAMPLES

Example 1: In the following example, the ift command reads the User Flaw table from device 0101, and it writes the nonslippable flaws to the ASCII Flaw table /etc/aft/0101.

```
ift /dev/ift/0101 > /etc/aft/0101
```

Example 2: In the following example, if device 0130.4 is an array, the flaw tables from drives 1 and 3 will be the only ones read.

```
ift -s 012 /dev/ift/0130.4
```

SEE ALSO

dsk(4) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

General UNICOS System Administration, Cray Research publication SG–2301

NAME

inetd – Performs Internet super-server function

SYNOPSIS

/etc/inetd [-d] [-R *rate*] [*configuration file*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The inetd command, invoked at boot time by /etc/sdaemon, which is invoked by /etc/tcpstart, which is invoked by netstart(8), listens for connections on certain Internet sockets, and when a connection is found on one of its sockets, it decides to which service the socket corresponds and invokes a program to service the request. After the program is finished, inetd continues to listen on the socket (except in selected cases; see the inetd.conf(5) man page for more information). Essentially, inetd allows running one daemon to invoke several others, reducing load on the system.

The inetd command accepts the following options:

-d — Turns on debugging for connection-oriented services.

-R *rate* — Specifies the maximum number of times a service can be invoked in one minute; the default is 90.

configuration file — File from which, on execution, inetd reads its configuration information. This file is by default, /etc/inetd.conf. The configuration file contains the following information about each server: service name, socket type, protocol, wait/nowait, user, server program, and server program arguments. For RPC-based servers, protocol version numbers are also specified. See inetd.conf(5) for more information about format and content.

The inetd command provides several trivial services internally by use of routines within itself. These services are echo, discard, chargen (character generator), daytime (human readable time), and time (machine readable time, in the form of the number of seconds since midnight, January 1, 1900). For details of these services, consult the relevant RFCs, as follows:

echo — RFC 862, *Echo Protocol*, Postel, J. B., May 1983.

discard — RFC 863, *Discard Protocol*, Postel, J. B., May 1983.

chargen — RFC 864, *Character Generator Protocol*, Postel, J. B., May 1983.

daytime — RFC 867, *Daytime Protocol*, Postel, J. B., May 1983.

time — RFC 868, *Time Protocol*, Postel and Harrenstein, May 1983.

The `inetd` command rereads its configuration file when it receives a hangup signal, `SIGHUP`. Services may be added, deleted, or modified when the configuration file is reread. `inetd` also reregisters all of its RPC-based servers with `portmap` at this time.

`inetd` invokes the server process at the security label of the incoming connection or datagram.

ERROR MESSAGES

The `inetd` server logs error messages using `syslog`(3C). Important error messages and their related explanations are as follows:

Error Message	Explanation
`server failing (> 90 connections in 60 seconds), service terminated`	The number of requests for the specified server in the past minute has exceeded the limit. The limit exists to prevent a broken program or a malicious user from swamping the system. This message may occur for any of the following reasons: 1. A number of hosts are requesting the service in a short period of time 2. A "broken" client program is requesting the service too frequently in a short period of time 3. A malicious user is running a program to invoke the service in a "denial of service" attack 4. The invoked service program has an error that causes clients to retry too quickly To change the rate limit, use the [`-R`] option, as described above. Once the limit has been reached, the service will be reenabled automatically following a five minute wait period.
`No such user` *user*`, service ignored`	No entry for *user* exists in the `passwd` file. This message occurs when `inetd` rereads the `configuration` file.
`getpwnam:` *user:* `No such user`	No entry for *user* exists in the `passwd` file. This message occurs when the service is invoked.
`can't set uid number`	The user ID for the entry's user is invalid.
`can't set gid number`	The group ID for the entry's user is invalid.

NOTES

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

Active Category	Action
`system`, `secadm`, `sysadm`	Allowed to use this command.

If the `PRIV_SU` configuration option is enabled, the super user is allowed to use this command.

FILES

`/etc/inetd.conf` Default configuration file for `inetd`

SEE ALSO

`ftpd`(8), `netstart`(8), `rexecd`(8), `rlogind`(8), `rshd`(8), `telnetd`(8), `tftpd`(8) in the *UNICOS Administrator Commands Reference Manual*, Cray Research publication SR–2022

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

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

UNICOS Networking Facilities Administrator's Guide, Cray Research publication SG–2304

NAME

infd – Allows the operator to display informative messages

SYNOPSIS

/usr/lib/msg/infd

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The infd command allows the operator to display informative messages. The operator does not reply to informative messages, and the informative message is removed after the operator has seen it. infd displays messages in the order in which it receives them.

NOTES

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

Privilege Text	Action
showall	Messages are not subject to security label restrictions.
both	Messages are not subject to security label restrictions.

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

Active Category	Action
system, secadm, sysadm, sysops	Allowed to use this command to view all messages.

If the PRIV_SU configuration option is enabled, the super user is allowed to use this command to view all messages.

SEE ALSO

msgdaemon(8), msgdstop(8), rep(8)

msgi(1), msgr(1), privtext(1) in the *UNICOS User Commands Reference Manual*, Cray Research publication SR–2011

NAME

infocmp – Compares or displays terminfo(5) descriptions

SYNOPSIS

/usr/bin/infocmp [-c] [-d] [-n] [-C] [-I] [-L] [-r] [-u] [-s *sort_option*] [-v] [-V] [-w *width*] [-1] [-A *directory*] [-B *directory*] [*termnames*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The infocmp command can be used to compare a binary terminfo(5) entry with other terminfo(5) entries, to rewrite a terminfo(5) description to take advantage of the terminfo(5) string capability use, or to display a terminfo(5) description from the binary file (term(5)) in a variety of formats. In all cases, the Boolean fields are displayed first, followed by the numeric fields, followed by the string fields.

Default Options

If no options are specified and zero or one *termnames* are specified, the -I option is assumed. If more than one *termname* is specified, the -d option is assumed.

Comparison Options

The infocmp command compares the terminfo(5) description of the first terminal *termname* with each of the descriptions given by the entries for the other terminals' *termnames*. If a capability is defined for only one of the terminals, the value returned depends on the type of the capability: F for Boolean variables, -1 for integer variables, and NULL for string variables.

-c Produces a list of each capability that is common between the two entries. Capabilities that are not set are ignored. This option can be used as a quick check to see whether the -u option is worth using.

-d Produces a list of each capability that is different. In this manner, if there are two entries for the same terminal or similar terminals, using infocmp shows what is different between the two entries. This is sometimes necessary when more than one person produces an entry for the same terminal and you want to see what is different between the two.

-n Produces a list of each capability that is in neither entry. If no *termnames* are given, the environment variable TERM is used for both of the *termnames*. This can be used as a quick check to see whether anything was left out of the description.

Source Listing Options

The -C, -I, and -L options produce a source listing for each terminal named.

-C Uses the termcap names.

-I Uses the terminfo(5) names.

`-L` Uses the long C variable name listed in the include file <`term.h`>.

`-r` When using `-C`, displays all capabilities in `termcap` form.

If no *termnames* are given, the `TERM` environment variable is used for the terminal name.

The source code produced by the `-C` option may be used directly as a `termcap` entry, but not all of the parameterized strings may be changed to the `termcap` format. `infocmp` attempts to convert most of the parameterized information, but unconverted information is plainly marked in the output and commented out. These strings should be edited by hand.

All padding information for strings is collected and placed at the beginning of the string, where `termcap` expects it. Mandatory padding (padding information with a trailing '/') becomes optional.

All `termcap` variables that are no longer supported by `terminfo`(5), but can be derived from other `terminfo`(5) variables, are displayed. Not all `terminfo`(5) capabilities are translated; only the variables that were part of `termcap` are normally displayed. Specifying the `-r` option removes this restriction, allowing all capabilities to be displayed in `termcap` form.

It is not always possible to convert a `terminfo`(5) string capability into an equivalent `termcap` format, because padding is collected to the beginning of the capability, not all capabilities are displayed, mandatory padding is not supported, and `termcap` strings are not as flexible. Subsequently converting the `termcap` file back into `terminfo`(5) format does not necessarily reproduce the original `terminfo`(5) source.

The following table shows some common `terminfo`(5) parameter sequences, their `termcap` equivalents, and some terminal types that commonly have such sequences:

`terminfo`	`termcap`	Representative terminals
%p1%c	%.	adm
%p1%d	%d	hp, ANSI standard, vt100
%p1%'x'%+%c	%+x	concept
%i	%i	ANSI standard, vt100
%p1%?%'x'%>%t%p1%'y'%+%;	%>xy	concept
%p2 is displayed before %p1	%r	hp

`use` String Capability Option

`-u` Produces a `terminfo`(5) source description of the first terminal *termname* that is relative to the sum of the descriptions given by the entries for the other terminals' *termnames*. It does this by analyzing the differences between the first *termname* and the other *termnames* and producing a description with the string capability `use` for the other terminals. In this manner, it is possible to retrofit generic `terminfo`(5) entries into a terminal's description. If two similar terminals exist but were coded at different times or by different people so that each description is a full description, using `infocmp` shows what can be done to change one description so that it is relative to the other.

A capability is displayed with an at-sign (@) if it no longer exists in the first *termname* but one of the other *termname* entries contains a value for it. A capability's value is displayed if the value in the first *termname* is not found in any of the other *termname* entries, or if the first of the other *termname* entries that has this capability specifies a different value for the capability than that in the first *termname*.

The order of the other *termname* entries is significant. Because the `terminfo`(5) compiler `tic`(8) does a left-to-right scan of the capabilities, specifying two `use` entries that contain differing entries for the same capabilities produces different results, depending on the order in which the entries are given. `infocmp` flags any such inconsistencies between the other *termname* entries as they are found.

Alternatively, specifying a capability after a `use` entry that contains that capability causes the second specification to be ignored. Using `infocmp` to recreate a description can be a useful check to ensure that everything was specified correctly in the original source description.

Another error that does not cause incorrect compiled files, but slows down the compilation time, is specifying extra `use` string capabilities that are superfluous. `infocmp` flags any other *termname* `use` entries that are not needed.

Other Options

`-s` *sort_options* Sorts the fields within each type. The `-s` option accepts the following arguments:

- `d` Leaves fields in the order in which they are stored in the `terminfo`(5) database
- `i` Sorts by `terminfo`(5) name
- `l` Sorts by the long C variable name
- `c` Sorts by the `termcap` name

If an `-s` option is not given, the fields displayed are sorted alphabetically by the `terminfo`(5) name within each type, except in the case of the `-C` or the `-L` options, which cause the sorting to be done by the `termcap` name or the long C variable name, respectively.

`-v` Prints out tracing information on standard error as the program runs.

`-V` Prints out the version of the program in use on standard error and exit.

`-w` *width* Changes the output to *width* characters.

`-1` Causes the fields to be displayed one to a line. Otherwise, the fields are displayed several to a line to a maximum width of 60 characters.

Change Databases Options

The location of the compiled `terminfo`(5) database is taken from the environment variable `TERMINFO`. If the variable is not defined, or if the terminal is not found in that location, the system `terminfo`(5) database, usually in `/usr/lib/terminfo`, is used. The `-A` and `-B` options may be used to override this location.

`-A` *directory* Sets `TERMINFO` for the first *termname*.

`-B` *directory* Sets `TERMINFO` for the other *termnames*.

With this, it is possible to compare descriptions for a terminal with the same name located in two different databases. This is useful for comparing descriptions for the same terminal created by different people. Otherwise, the terminals would have to be named differently in the `terminfo`(5) database for a comparison to be made.

ENVIRONMENT VARIABLES

`TERM` Used by screen editors and other screen-based programs to identify the terminal type in use. This variable need not be set if you are not going to use `vi`(1) or `more`(1). `TERM` maps the terminal's keyboard and screen characteristics to a `TERMINFO` definition. The value of `TERM` must be one of the known terminal definition names. These names are found in the `/usr/lib/terminfo` directory, which is organized into 36 directories (0 through 9 and a through z, as well as A through Z if necessary). Each terminal definition is kept in the directory that matches the first character of the terminal definition name.

`TERMINFO` Used to identify the path to an alternative terminal information (`TERMINFO`) directory. The `/usr/lib/terminfo` default directory file contains terminal definitions. You can create your own terminal definitions, compile them, and store them in one of your own directories. Commonly, this directory is `TERMINFO=$HOME/terminfo`. For the system to recognize an alternative directory, you must then set and export `TERMINFO`.

MESSAGES

`malloc is out of space!`
Not enough memory was available to process all of the terminal descriptions requested. Run infocmp several times, each time including a subset of the desired `termnames`.

`use= order dependency found:`
A value specified in one relative terminal specification was different from that in another relative terminal specification.

`´use=term' did not add anything to the description.`
A relative terminal name did not contribute anything to the final description.

`must have at least two terminal names for a comparison to be done.`
The `-u`, `-d`, and `-c` options require at least two terminal names.

FILES

`/usr/lib/terminfo/?/*` Database of compiled terminal descriptions

SEE ALSO

`tic`(8)

`tput`(1), `tset`(1B) in the *UNICOS User Commands Reference Manual*, Cray Research publication SR–2011

`curses`(3) (available only online)

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

NAME

`init`, `telinit` – Controls process initialization

SYNOPSIS

`/etc/init` [`0123456SsQqM`]

`/etc/telinit` [`0123456sSQqabcM`]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `init` command is a general process spawner that primarily creates processes from a procedure stored in the `/etc/inittab` file (see `inittab`(5)). `init` controls the state (run level) of the system by controlling autonomous processes required by the various states.

`init` considers the system to be in a certain run level at any given time. You can think of a run level as a software configuration of the system in which each configuration allows only a selected group of processes to exist. The processes spawned by `init` for each of these run levels are defined in the `/etc/inittab` file. `init` can be in one of eight run levels, 0 through 6 and `S` or `s`. An appropriately authorized user can change the run level by running `/etc/init` (which is linked to `/etc/telinit`). This user-spawned `init` sends appropriate signals to the original `init` spawned by the operating system when the system was rebooted, telling the original `init` which run level to change.

`init` is invoked inside the UNICOS operating system as the last step in the boot procedure. It is always process ID 1, and all other processes are spawned, directly or indirectly, from it. `init` first checks to see if there was a run-level specified in the initial UNICOS exchange package. On CRAY Y-MP E series, this is set by the OWS program `mfstart`. If this value is nonzero, `init` uses it for the initial run level. `init` next looks for `/etc/inittab` to see whether there is an entry of the type *initdefault* (see `inittab`(5)). If such an entry exists, and no run level was set in the exchange package, `init` uses the run level specified in that entry as the initial run level to enter. If this entry is not in `/etc/inittab` or `/etc/inittab` is not found, `init` requests that you enter a run level from the system console, `/dev/console`.

If an `S` or `s` is entered, `init` goes into the single-user level, which is the only run level that does not require a properly formatted `/etc/inittab` file to exist. If `/etc/inittab` does not exist, by default the only legal run level that `init` can enter is the single-user level. In the single-user level, the console terminal `/dev/console` is opened for reading and writing. To exit from the single-user run level, one of two options can be selected. First, if the shell is terminated (using an end-of-file), `init` reprompts for a new run level. Second, `init` or `telinit` can signal `init` and force it to change the run level of the system.

When booting the system, failure of `init` to prompt for a new run level could be because the device `/dev/console` is missing.

When `init` prompts for the new run level, you can enter only one of the digits 0 through 6 or the letters `S` or `s`. If you enter a digit from 0 through 6, `init` enters the corresponding run level. Any other input is rejected, and you are prompted again. If this is the first time `init` has entered a run level other than single-user mode, `init` first scans `/etc/inittab` for special entries of the type *boot* and *bootwait*. These entries are invoked, provided that the run level entered matches that of the entry before any normal processing of `/etc/inittab` occurs. In this way, any special initialization of the operating system, such as mounting file systems, can occur before users are allowed onto the system. `init` scans the `/etc/inittab` file to find all entries that are to be processed for that run level.

Run level 2 is usually defined by the administrator to contain all the terminal processes and daemons that are spawned in the multiuser environment. In a multiuser environment, the `/etc/inittab` file is usually set up so that `init` creates a process for each terminal on the system.

For terminal processes, ultimately the shell will terminate because of an end-of-file either typed explicitly or generated as the result of hanging up. When `init` receives a child process termination signal, telling it that a process it spawned has died, `init` records the fact and the reason the process died in `/etc/utmp` and `/etc/wtmp`, if such a file exists (see `who`(1)). A history of the processes spawned is kept in `/etc/wtmp`, if such a file exists.

To spawn each process in the `/etc/inittab` file, `init` reads each entry, and for each entry that should be respawned, it forks a child process. After it has spawned all the processes specified by the `/etc/inittab` file, `init` waits for one of its descendant processes to die or until `init` is signaled by `init` or `telinit` to change the system's run level. When one of the preceding conditions occurs, `init` reexamines the `/etc/inittab` file. New entries can be added to the `/etc/inittab` file at any time; however, `init` still waits for one of the preceding three conditions to occur. To provide for an instantaneous response, the `init Q` or `init q` command can wake `init` to reexamine the `/etc/inittab` file.

When `init` is requested to change run levels (using `telinit`), `init` sends the warning signal `SIGTERM` to all processes that are undefined in the target run level. On IOS model E systems, `init` also sends an advisory O packet with the new run level to the OWS. `init` waits `TWARN` (10) seconds before forcibly terminating these processes by using the kill signal `SIGKILL`.

The `telinit` command, which is linked to `init`, directs the actions of `init`. It takes a one-character argument and signals `init` by using the kill system call to perform the appropriate action.

The following arguments serve as directives to `init`:

`0` through `6`	Places the system in one of the run levels 0 through 6.
`a`, `b`, `c`	Processes only those `/etc/inittab` file entries having the `a`, `b`, or `c` run level set.
`Q`, `q`	Reexamines the `/etc/inittab` file.
`S`, `s`	Enters the single-user environment.

`M` Enters maintenance run level.
Note: Reserved for use by Cray Research. See NOTES subsection.

Only an appropriately authorized user can run `telinit`.

NOTES

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

Active Category	Action
`system`, `secadm`, `sysadm`	Allowed to run these commands and to change run levels.

If the `PRIV_SU` configuration option is enabled, the super user is allowed to run these commands and to change run levels.

The maintenance run level (`M`) is similar to run levels 0 through 6, with the exception that the `bootwait` and `boot` functions are not performed if `M` is the intial run level. The maintenance run level is reserved for use by Cray Research; its functions may change without notice.

MESSAGES

If `init` finds that it is continuously respawning an entry from `/etc/inittab` more than 10 times in 2 minutes, it assumes that the command string contains an error and generates an error message on the system console; it refuses to respawn this entry until either 5 minutes has elapsed or it receives a signal from a user `init` (`telinit`). This prevents `init` from eating up system resources when someone makes a typographical error in the `/etc/inittab` file or when a program is removed that is referenced in the `/etc/inittab` file.

FILES

`/etc/inittab`

`/etc/utmp`

`/etc/wtmp`

SEE ALSO

`brc`(8), `cleantmp`(8), `getty`(8)

`login`(1), `sh`(1), `who`(1) in the *UNICOS User Commands Reference Manual*, Cray Research publication SR–2011

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

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

General UNICOS System Administration, Cray Research publication SG–2301

NAME

`initif` – Configures network interfaces

SYNOPSIS

`/etc/initif` [`-F` *family*] [`-f` *file*] [`-h` *header*] [`-q`] [*interfaces*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `initif` script configures network interfaces according to information contained in a tabular configuration file. Only the specified interfaces are configured. If no interfaces are specified, all interfaces listed in the configuration file will be configured.

When configuring interfaces, unless you use the `-q` option, `initif` prints a header line, which is not terminated by a new-line character, and prints the name of each interface successfully configured, terminating the list with a period and new-line character. The default header string is as follows:

`Configuring` *type* network interfaces:

type is a string that indicates the type of network interfaces being configured:

`all`	Default case of all listed interfaces.
`selected`	One or more interfaces have been specified on the command line.
`TCP/IP`	For all interfaces in the `inet` address family.

The following options are accepted by `initif`:

`-F` *family*	Specifies that only interfaces of address type *family* will be configured. *initif* will configure up only interfaces whose *family* field matches the specified *family*. (See the File Format subsection.) The match may be case-insensitive; that is, a specification of `-F INET` on the command line matches `inet` entries in the configuration file.
`-f` *file*	Specifies *file* as the configuration file from which to fetch information about interfaces to be configured. If `-f` is not specified, the default configuration file is used.
`-h` *header*	Specifies the string *header*, rather than the default, as the header printed before the list of interfaces configured. White space in a header string must be escaped from the shell.
`-q`	Specifies quiet mode; prints neither a header nor the names of interfaces configured.
interfaces	Specifies interfaces to be configured.

File Format

A configuration file for `initif` consists of a series of lines with the following format:

name hyfile family address destination arguments ...

The elements in an entry have the following meanings:

name Name of the interface (for example, `hy0`).

hyfile Name of the file that contains the hardware address-mapping information to be attached to the interface through the `hyroute`(8) command. A *hyfile* of `-` (one hyphen) indicates that no hardware address-mapping information should be attached to this interface. If *hyfile* begins with a leading slash (/), it is assumed to be the full path name of the file; otherwise, the file name is interpreted relative to the UNICOS configuration directory `/etc/config`.

family Address family to be used for the interface (typically `inet`).

address Network address to be associated with the interface.

destination Network address of the destination address if this interface is a point-to-point link. A *destination* of `-` (one hyphen) indicates that the interface is not a point-to-point link.

arguments Keyword-value pairs of arguments (for example, `netmask 0xffffff00 iftype hy`) to be passed to `ifconfig`(8) when bringing up the interface; see `ifconfig`(8) for a complete list of arguments.

An initial # character on a line indicates that the line is a comment.

MESSAGES

If interface *name* is not listed in configuration file *file*, the `initif` script issues the following message.

`initif: interface ‘`*name*`‘ not found in ‘`*file*`‘`

FILES

`/etc/config/interfaces` Default configuration file

SEE ALSO

`hyroute`(8), `ifconfig`(8)