NAME

`intro` – Introduces system maintenance commands, network maintenance and operation commands, and application programs that invoke shell procedures

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

This section describes, in alphabetical order, commands that are used mainly for system maintenance and administration purposes for all Cray Research systems.

The following terms identify UNICOS command components:

Command Component	Definition
Command	Name of an executable file.
Option	Command-line element indicated by a hyphen, followed by a letter.
Option-argument	Character string that supplies information for the preceding option.
Operand	Command-line element to be passed to the command; not associated with an option.

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

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

EXIT STATUS

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

BUGS

Many commands do not use the aforementioned syntax.

SEE ALSO

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

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

UNICOS Command Conventions, Cray Research publication CP–2058

NAME

acct – Overview of standard UNIX System V accounting commands

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The UNICOS operating system supports two accounting packages, Cray Research system accounting (CSA) and standard UNIX System V accounting. The standard UNIX accounting package is a set of C programs and shell scripts that provides methods for collecting resource use data per process, recording connect sessions, monitoring disk usage, and charging fees to specific logins. This man page describes the accounting commands used by either accounting system. The acctsh(8) man page describes the shell scripts used by either accounting system.

The UNICOS kernel performs process accounting. On termination of a process, one record per process is written to a file, usually /usr/adm/acct/day/pacct. The acctprc1 and acctprc2 commands (see acctprc(8)) summarize this data for charging purposes; the acctcms(8) command summarizes command usage, and the acctcom(1) command reports current process data.

Various programs handle connect-time accounting by writing records into the /etc/wtmp file, as described in utmp(5). The programs described on the acctcon(8) man page convert this file into login session and charging records, which can then be summarized by the acctmerg(8) command.

Process accounting, connect-time accounting, and any accounting records in the format described on the acct(5) man page can be merged and summarized into total accounting records by acctmerg (see the acct(5) for the tacct format). The prtacct(8) command formats or prints any tacct accounting records.

FILES

/etc/udb	User validation file that contains user control limits and contains login name to user ID conversions.
/etc/wtmp	Contains login and logoff history information.
/usr/adm/acct/day/pacct	Contains current process accounting information.
/usr/lib/acct	Contains most of the accounting commands listed in this manual.

SEE ALSO

acctcms(8), acctcon(8), acctdisk(8), acctdusg(8), acctmerg(8), accton(8), acctprc(8), acctsh(8), acctwtmp(8), csa(8), diskusg(8), dodisk(8), fwtmp(8), lastlogin(8), monacct(8), nulladm(8), prctmp(8), prdaily(8), prtacct(8), remove(8), runacct(8), shutacct(8), startup(8), turnacct(8)

acctcom(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

UNICOS Resource Administration, Cray Research publication SG–2302

NAME

acctcms – Summarizes command usage from per-process accounting records

SYNOPSIS

/usr/lib/acct/acctcms [-a [[[-p] [-o]] [-e]]] [-c] [-j] [-n] [-s] [-S [-A]] *files*

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The acctcms command reads one or more *files*, usually in the format described in acct(5). It adds all records for processes that executed identically named commands, and it sorts and writes them to the standard output, usually using an internal summary format. acctcms accepts the following options:

-a Prints output in ASCII rather than in the internal summary format. You can use the following options only with the -a option:

- -p Outputs a prime-time-only command summary.
- -o Outputs a nonprime-time-only (offshift) command summary.
- -e Outputs an extended report, printing additional fields. You can use the -e option only when the -p or -o options also are selected with the -a option.

  The default output produced with the -a option includes command name, number of times executed, total kcore-minutes, total CPU minutes, total real minutes, mean size (in K), mean CPU minutes per invocation, the amount of CPU resources (proportional to other processes) used, characters transferred, and blocks read and written, as in acctcom(1). Usually, the output is sorted by total kcore-minutes.

  When you specify both -p and -o with -a, acctcms produces a combination prime and nonprime time report.

  All output summaries indicate octal usage except number of times executed, CPU minutes, and real minutes, which are split into prime and nonprime.

-c Sorts by total CPU time rather than total kcore-minutes.

-j Combines all commands invoked only once under ‘‘***other’’.

-n Sorts by number of command invocations.

-s Indicates that any file names encountered hereafter are already in internal summary format.

-S Indicates that the Session record format is used on input.

-A Causes all jobs (even nonterminated sessions) to be considered. You must use this option with the -S option.

EXAMPLES

A typical sequence for performing daily command accounting and for maintaining a running total is as follows:

```
acctcms file . . . >today
cp total previoustotal
acctcms -s today previoustotal >total
acctcms -a -s today
```

SEE ALSO

acct(8), acctcon(8), acctmerg(8), acctprc(8), acctsh(8), fwtmp(8), runacct(8)

acctcom(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

acctcon1, acctcon2 – Performs connect-time accounting

SYNOPSIS

/usr/lib/acct/acctcon1 [-l *file*] [-o *file*] [-p] [-t]

/usr/lib/acct/acctcon2

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The acctcon1 command converts a sequence of login and logoff records read from its standard input to a sequence of records, one per login session. Usually, its input should be redirected from the /etc/wtmp file. Its output is ASCII, and it specifies device, user ID, login name, prime connect time (seconds), nonprime connect time (seconds), session starting time (numeric), and starting date and time.

The acctcon1 command maintains a list of lines cataloging which users are logged in. When it reaches the end of its input, it emits a session record for each line that still appears to be active. Usually, it assumes that its input is a current file; therefore, it uses the current time as the ending time for each session still in progress.

The acctcon1 command accepts the following options:

-l *file* Creates *file* to contain a summary of line usage that shows line name, number of minutes used, percentage of total elapsed time used, number of sessions charged, number of logins, and number of logoffs. This file helps track line usage, identify bad lines, and find software and hardware inconsistencies and errors. Hang-up, termination of login(1), and termination of the login shell each generate log-off records; therefore, the number of logoffs is often three to four times the number of sessions. See init(8) and utmp(5).

-o *file* Fills *file* with an overall record for the accounting period, giving starting time, ending time, number of reboots, and number of date changes.

-p Prints input only, showing line name, login name, and time (in both numeric and date/time formats).

-t Uses, rather then the current time, the last time found in its input as the ending time for active sessions. This ensures reasonable and repeatable numbers for noncurrent files.

acctcon2 expects as input a sequence of login session records and converts them into total accounting records (see tacct format in acct(5)).

BUGS

The line-usage report is confused by date changes. Use `wtmpfix` (see `fwtmp`(8)) to correct this situation.

EXAMPLES

Typically, these commands are used as follows (the `ctmp` file is created only for the use of `acctprc`(8) commands):

```
acctcon1 -t -l lineuse -o reboots <wtmp | sort  +1n  +2 >ctmp
acctcon2 <ctmp | acctmerg >ctacct
```

FILES

`/etc/wtmp`

SEE ALSO

`acct`(8), `acctcms`(8), `acctmerg`(8), `acctprc`(8), `acctsh`(8), `fwtmp`(8), `init`(8), `runacct`(8)

`acctcom`(1), `login`(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

acctdisk – Converts disk data to cacct or tacct format

SYNOPSIS

/usr/lib/acct/acctdisk [-a] [-A]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

By default, the acctdisk command reads standard input and converts records to tacct format, which it writes to standard output. (See acct(5) for the format.) Each input record contains a user ID, a login name, and the number of disk blocks allocated. The input file is generally the default ouput from the diskusg(8) command. You then can merge the tacct records with other tacct records by using the acctmerg(8) command.

The -a and -A options convert input records to cacct format, which is used by the Cray Research system accounting (CSA) feature. These records can be merged with other cacct records by using the csaaddc(8) command.

The acctdisk command accepts the following options:

-a Accepts as input the output produced by the diskusg(8) command specified with the -a option and produces output in cacct format.

-A Accepts as input the output produced by the diskusg(8) command specified with the -A output and produces output in cacct format.

EXAMPLES

The following example displays the conversion of diskusg(8) output to tacct format:

```
/usr/lib/acct/diskusg /dev/dsk/tmp | /usr/lib/acct/acctdisk > tacctfile
```

The following example displays the conversion of the contents of the diskdata file, the output of diskusg -a /dev/dsk/tmp, to cacct format:

```
/usr/lib/acct/acctdisk -a < diskdata > cacctfile
```

FILES

/etc/udb User validation file that contains user control limits; used for user information.

SEE ALSO

`acct`(8), `acctmerg`(8), `acctsh`(8), `csa`(8), `csaaddc`(8), `csaperiod`(8), `diskusg`(8), `runacct`(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

`acctdusg` – Computes and displays disk resource consumption by login

SYNOPSIS

`/usr/lib/acct/acctdusg` [`-p` *path*] [`-u` *file*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `acctdusg` command reads its standard input, usually from the `find / -print` command, and it computes disk resource consumption, including indirect blocks, by login. Output is written to standard output.

The `acctdusg` command accepts the following options:

`-p` *path* Specifies the path name of the user database (UDB) file. By default, the UDB file is defined as `/etc/udb`.

`-u` *file* Writes the names of files that have not been charged to anyone to the specified *file*. This information can help you identify users who are trying to avoid disk charges.

EXAMPLES

In the following example, `acctdusg` displays disk resource consumption by login:

```
find / -print | /usr/lib/acct/acctdusg > dusgdata
```

FILES

`/etc/udb` User validation file that contains user control limits; used for user information.

SEE ALSO

`acct`(8), `acctsh`(8)

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

`udb`(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

acctmerg – Merges or adds total accounting files

SYNOPSIS

/usr/lib/acct/acctmerg [-a [-b] [-c] [-d] [-f] [-h] [-j] [-m] [-M] [-n] [-w] [-x] [-y]] [-i] [-p] [-s] [-t] [-v] [*files*]

/usr/lib/acct/acctmerg [-a [-b] [-c] [-d] [-f] [-h] [-j] [-m] [-M] [-n] [-w] [-x] [-y]] [-i] [-p] [-s] [-u] [-v] [*files*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The acctmerg command reads its standard input and additional files, all in the tacct format (see acct(5)) or an ASCII version thereof. It merges these inputs by adding records with keys (usually, user ID, login name, and account name) that are identical, and it expects the inputs to be sorted on those keys.

The acctmerg command accepts the following options:

-a Produces output in an ASCII version of tacct. The following options are valid only if you specify the -a option. They specify a subset of the total information and allow a flexible control of the resulting output format. If you specify any of the following options, each output line includes account name, user ID, and login name:

- -b Specifies system billing information (SBUs) and operational fees.
- -c Specifies CPU time memory integral and connect time in seconds.
- -d Specifies cumulative disk-block-usage count and number of disk samples taken.
- -f Provides full information about each entry. The output format consists of several lines per entry and provides all of the preceding information plus additional data about device-specific I/O (if available).
- -h Writes information header for all requested fields.
- -j Specifies number of processes and number of jobs.
- -m Specifies user CPU time breakdown for multiple CPUs running in parallel (multitasked processes only).
- -M (Cray MPP systems only) Specifies Cray MPP information.
- -n Splits data into prime and nonprime time data (two lines of output for each entry).
- -w Specifies I/O wait time and I/O wait-time memory integral in seconds.
- -x Specifies number of blocks transferred; real and logical I/O request counts.

-y Specifies number of SDS blocks transferred.

-i Specifies that input files are in an ASCII version of tacct.

-p Prints input without processing.

-s Summarizes by account name rather than user ID, login name, and account name (cannot be used with the -u option). Input that was created by using the acctprc2 -s option (see acctprc(8)) must be processed with this option.

-t Produces one record that totals all input.

-u Provides summary by user IDs rather than user ID, login name, and account name (cannot be used with the -s option).

-v Produces output in verbose ASCII format, with more precise notation for floating-point numbers.

NOTES

When using the -v or -a options, several fields within tacct records are printed out in a slightly different order than is defined by the tacct structure. Instead of printing ta_dc, ta_pc, and ta_sc, the order is switched to ta_pc, ta_sc, and ta_dc. If records are read in using the -i (ASCII input) option, the expected order of input differs from the tacct structure in exactly the same manner.

EXAMPLES

The following sequence is useful for making corrections to any file kept in this format:

```
acctmerg -v <file1 >file2
      edit file2 as desired . . .
acctmerg -i <file2 >file1
```

The following example merges three input files, ifile0, ifile1, and ifile2; the output file is ofile:

```
acctmerg ifile1 ifile2 < ifile0 > ofile
```

SEE ALSO

acct(8), acctcms(8), acctcon(8), acctprc(8), acctsh(8), fwtmp(8), runacct(8)

acctcom(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

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

NAME

accton – Controls process accounting

SYNOPSIS

/usr/lib/acct/accton [*file*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The accton command specified without operands turns off process accounting. To turn on process accounting, execute the accton command with the following operand:

file Name of the *file* to which the kernel appends process accounting records. (See acct(2) and acct(5)).

If the specified *file* does not exist, accton creates it and properly sets the owner group and mode of the file.

The super user or a user who is in the group adm and has permission bit acct set in their user database (UDB) entry (see udbgen(8)) must invoke accton.

NOTES

The accton command is rarely invoked alone. Use the turnacct(8) command to enable and disable processing accounting.

Sites may allow users in the group adm who have the permission bit acct set in their UDB entries to run Cray Research system accounting (CSA). However, such users cannot run accounting after a super user has done so, because the group ID and permissions of the files will have changed. In this case, the csaperm(8) command must be executed to reset group IDs and permissions before nonsuper users can run accounting.

FILES

/etc/udb	User validation file that contains user control limits; used for user information.
/usr/lib/acct/day/pacct	Contains current process accounting information.

SEE ALSO

`acct`(8), `acctsh`(8), `csaperm`(8), `turnacct`(8), `udbgen`(8)

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

`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

acctprc1, acctprc2 – Processes accounting

SYNOPSIS

```
/usr/lib/acct/acctprc1 [ctmp]
/usr/lib/acct/acctprc2 [-s]
```

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The acctprc1 command reads input in the form described by acct(5), adds login names that correspond to user IDs, then writes per-process temporary data records (in ptmp format) to standard output.

The acctprc1 command accepts the following option:

ctmp If you specify the ctmp file, it is expected to contain a list of login sessions, in the form described in acctcon(8), sorted by user ID and login name. If you omit ctmp, it obtains login names from the user database file (/etc/udb). The information in ctmp helps it distinguish among different login names that share the same user ID.

The acctprc2 command reads records in the form written by acctprc1, summarizes them by user ID and name, then writes the sorted summaries to the standard output as total accounting records (tacct format).

The acctprc2 command accepts the following option:

-s Sorts the output by ascending account ID and by ascending user ID. If this output is processed later by acctmerg(8), you must use the acctmerg -s option.

Typically, these commands are used as follows:

```
acctprc1 ctmp </usr/adm/acct/day/pacct | acctprc2 >ptacct
```

BUGS

Although it is possible to distinguish among login names that share user IDs for commands run normally, it is difficult to do this for those commands run from cron(8), for example. You can do more precise conversion by faking login sessions on the console, using the acctwtmp(8) program.

The size of some structures in these two commands may be too large for small-memory machines. The dimensions for these structures are the A_SSIZE and A_USIZE variables, the maximum number of sessions and distinct login names per accounting run, respectively; they are defined in the /etc/config/acct_config file. To resize the structures to fit machine memory, decrease the values of these variables and rerun the commands.

FILES

`/etc/config/acct_config`	Contains configurable parameters.
`/etc/udb`	User validation file that contains user control limits and contains login names for system users.
`/usr/adm/acct/day/pacct`	Contains process accounting information.

SEE ALSO

`acct`(8), `acctcms`(8), `acctcon`(8), `acctmerg`(8), `acctsh`(8), `acctwtmp`(8), `cron`(8), `fwtmp`(8), `runacct`(8)

`acctcom`(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

acctsh – Overview of accounting shell scripts

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The UNICOS operating system supports two accounting packages, Cray Research system accounting (CSA) and standard UNIX System V accounting. Both packages consist of a set of C programs and shell scripts. Some shell scripts are used by both packages; others are unique to one package or the other.

Each shell script is described in detail on a separate man page in the *UNICOS Administrator Commands Reference Manual*, Cray Research publication SR–2022. The following list contains the accounting shell scripts and their descriptions:

Script	Description
acct2csa	Converts standard UNIX System V accounting tacct files to CSA cacct format.
ckdacct	Checks the size of the daemon accounting files.
ckpacct	Checks the size of the process accounting files.
csaperiod	Performs CSA periodic accounting.
csarun	Processes the CSA daily accounting files and generates reports.
dodisk	Performs disk accounting.
lastlogin	Determines the last date on which each user logged in.
monacct	Performs UNIX System V monthly accounting.
nulladm	Creates an empty file with mode 664 and both, owner and group set to adm.
prctmp	Prints the UNIX System V accounting login session file.
prdaily	Prints the UNIX System V accounting daily report.
prtacct	Prints the UNIX System V accounting total accounting (tacct) file.
remove	Removes the temporary UNIX System V accounting files.
runacct	Processes the UNIX System V accounting daily accounting.
shutacct	Turns off process and daemon accounting.
startup	Turns on system accounting and daemon accounting.
turnacct	Turns process accounting on and off or switches accounting files.
turndacct	Turns daemon accounting on and off or switches accounting files.

FILES

/etc/config/acct_config	Accounting configuration file
/etc/wtmp	Login and logoff summaries
/usr/adm/acct/day	Directory that contains the current process and daemon accounting files
/usr/adm/acct/day/fee	UNIX System V fee accumulator
/usr/adm/acct/day/nqacct*	CSA Network Queuing System (NQS) accounting files
/usr/adm/acct/day/pacct	Current process accounting file
/usr/adm/acct/day/pacct∗	Unprocessed process accounting files
/usr/adm/acct/day/tpacct*	CSA tape accounting files
/usr/adm/acct/nite	Working directory
/usr/lib/acct/ptecms.awk	UNIX System V accounting shell script that generates exceptional usage by command name
/usr/lib/acct/ptelus.awk	UNIX System V accounting shell script that generates exceptional usage by login ID
/usr/adm/acct/sum	Summary directory
/usr/adm/acct/sum/tacct∗	Standard UNIX System V accounting total accounting files

SEE ALSO

acct(8), acctcon(8), ckdacct(8), ckpacct(8), csa(8), csaperiod(8), csarun(8), dodisk(8), lastlogin(8), monacct(8), nulladm(8), prctmp(8), prdaily(8), prtacct(8), remove(8), runacct(8), shutacct(8), startup(8), turnacct(8), turndacct(8)

UNICOS Resource Administration, Cray Research publication SG–2302

NAME

acctwtmp – Creates a utmp(5) record

SYNOPSIS

/usr/lib/acct/acctwtmp "*reason*"

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The acctwtmp command sends a utmp(5) record to standard output. The record contains the current time and a character string that describes the specified *reason*. acctwtmp assigns a record type of ACCOUNTING (see utmp(5)).

The acctwtmp command accepts the following operand:

"*reason*" Must be a string that consists of 11 or fewer characters, numbers, $, or spaces contained in double quotation marks.

EXAMPLES

The following examples are suggestions for using acctwtmp in reboot and shutdown procedures, respectively:

```
/usr/lib/acct/acctwtmp "acct on" >> /etc/wtmp
/usr/lib/acct/acctwtmp "acct off" >> /etc/wtmp
```

FILES

/etc/wtmp Contains login and logoff information

SEE ALSO

acct(8), acctsh(8), fwtmp(8), runacct(8), shutacct(8), startup(8)

utmp(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

aircksconf – Prints and validates an AIR configuration file

SYNOPSIS

```
/usr/air/bin/aircksconf [-D #] [-a] filename
/usr/air/bin/aircksconf -v [-D #] [-a] filename
/usr/air/bin/aircksconf -p [-D #] [-a] filename
/usr/air/bin/aircksconf -s [-D #] [-a] filename
```

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The aircksconf command reads the file specified on the command line and verifies that it is a valid automated incident reporting (AIR) configuration file. You can use this command to debug configuration files before using them with the rest of the AIR system. By default, the file is verified and a summary of the data is displayed.

The aircksconf command accepts the following options:

-D # Debug mode. Takes a small integer argument that specifies the number of debugging messages to print (the larger the number, the more messages). The range of most commands is 0 through 20.

-a Suppresses access checks. Prevents access checking of files named in the configuration. Useful at early debugging stages when files do not yet exist.

-v Skip validation mode. Skips validation; attempts to reprint configuration data.

-p Pretty print mode. Useful for reformatting the configuration file.

-s Silent mode. Only error messages are displayed.

MESSAGES

Many messages can be generated from this program. Most messages pertain to configuration file errors and are easy to interpret.

BUGS

File access checking messages might state only that the file is "not accessible" when the real problem could be file execute permissions.

SEE ALSO

aird(8)

NAME

aird – Automated incident reporting (AIR) daemon

SYNOPSIS

aird [-d] [-C *number*] [-D *number*] [-L *number*] *config_file*

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The aird daemon executes any configured monitoring functions. When the executed monitoring functions return, aird logs pertinent data into its binary log file. The report generators use this file as input, producing AIR reports on system and product availability.

When initiated, aird translates the contents of the specified configuration file into its internal processing worklist, which consists of the configured monitoring functions and their execution rates. aird also sets the specified return types in the environment. The monitoring functions use the environment variables as exit status values in order for the monitoring functions and aird to communicate.

If it receives a SIGHUP signal, aird stops executing the monitoring functions and rereads the configuration file. This capability allows for dynamic redefinition of aird's processing.

The aird daemon accepts the following options (the daily execution of aird is probably with no options):

-d Indicates that the process should not attempt to become a daemon. Use this option only for testing purposes.

-C *number* Specifies the time conversion rate factor. The specified integer value, *number*, indicates to aird how it should interpret the time specifications within the configuration file. The conversion factor is the number of seconds in a minute (the default is 60); by default, time in the configuration file is specified in minutes. You can compress AIR's internal time by lowering the conversion factor. For example, if you specify 1, time in the configuration file would be specified in units of seconds rather than minutes. Use this option only for testing purposes.

-D *number* Sets the debugging level. The *number* value is the number of debugging messages to print; possible values are integers in the range 1 through 20. A value of 0 sets debugging to off. As the number increases, so does the number of messages.

-L *number* Sets the logging level. Possible values are integers in the range 1 through 20. A value of 0 sets debugging to off. Higher numbers indicate more logging.

config_file AIR configuration file name.

BUGS

If the configuration file specified on the command line is not valid, the `aird` process terminates when it discovers errors in the file and ends the current session. Therefore, always run the `aircconf`(8) command on a new or edited configuration file before using that file.

EXAMPLES

In this example, the `aird` process is running in testing mode. The `-C` option, specified with a value of `1`, indicates that the unit of time in the configuration file should be interpreted as seconds. The `-d` option indicates that the program should not disconnect from the controlling terminal. The configuration file is `/etc/config_file`, and the messages are directed to the `/tmp/airlog` file.

```
aird -C 1 -d /etc/config_file >/tmp/airlog
```

SEE ALSO

`aircconf`(8)

UNICOS Resource Administration, Cray Research publication SG–2302

NAME

airdchk – Ensures that the AIR daemon (aird(8)) is running (cron(8) script)

SYNOPSIS

```
/usr/air/bin/airdchk
```

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The airdchk script executes as a cron(8) job and verifies that the automated incident reporting (AIR) coordinator daemon (aird(8)) is running. The script uses the airexist(8) command to verify that the process aird(8) exists on the system.

If the return from the airexist(8) invocation indicates that the process no longer exists on the system, mail is sent to root indicating that the aird(8) process is not running.

SEE ALSO

aird(8), airexist(8), cron(8)

NAME

airdet – Generates detailed AIR reports based on aird(8) binary log file

SYNOPSIS

airdet [-b *time*] [-e *time*] [-p *product*] [-f *function*] [-n *time*] [-T *types*] [-O] [-m] [-h] [-t] [-l] [-D *number*] [-w] *file1* [*file2* ...]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The airdet command prints the contents of the automated incident reporting (AIR) daemon's (aird(8)) binary log file, excluding the configuration header records. The airprconf(8) command prints the configuration header records; airdet prints all other records.

The aird(8) process logs three types of records into the binary log file. *Event records* comprise the majority of the records and are logged when each monitoring function is returned. These records indicate the product monitored by the function, the function's name, its start and end times, and the status of the product.

The aird(8) process also logs its own heartbeat record at the specified rate, as defined in the configuration file. These records are used to determine the availability of the AIR system; that is, the percent of time during a specified interval that aird(8) was running.

If a monitoring function does not return in the length of time specified in the configuration file, the aird(8) process kills the function and logs a time-out record, which indicates abnormal termination of the function.

The airdet command accepts the following options, which are listed in the following two groups:

- Selection options that specify the records selected for display
- Information options that specify the information to display for each of the selected records

By default, this command selects all of the event, heartbeat, and time-out records from the binary log file and prints the product and function names and the exit status types.

Selection Options

-b *time* Specifies the sample start time. Time format is as follows:

"[*month/day*[*/year*]] *hour:min*[:*sec*]"

-e *time* Specifies the sample end time. Time format is as follows:

"[*month/day*[*/year*]] *hour:min*[:*sec*]"

-p *product* Prints detailed records of only the specified product. An example of the many products provided, including user configurable products, is the Network Queuing System (NQS).

`-f` *function* Prints detailed records of only the specified function (for example, `existence`).

`-n` *time* Prints detailed records if their elapsed time is greater than the specified time.

`-T` *types* Prints detailed records of only the specified types (for example, `PROD_UNAVAILABLE`).

`-O` OR listed types to the `-T` option; AND is the default.

Information Options

`-m` Prints detailed message text.

`-h` Prints headers.

`-t` Prints time stamps.

`-l` Prints elapsed times.

`-D` *number* Sets debugging level. The *number* value is the number of debugging messages to print. Possible values are integers in the range 1 through 20. A value of `0` sets debugging to off. As the number increases, so does the number of messages.

`-w` Prints return value bits (in octal).

file1 [*file2* ...] `aird`(8) binary log files to use as input. One input file is required. You can specify others, separating the file names with spaces.

NOTES

If the date is not included in the time specifications for the `-b` and `-e` options, `airdet` uses today's date. Be careful when using the `-b` and `-e` options without date specifications if you are looking at binary log files no longer directly logged to by `aird`(8). It is possible that no records will be selected because the contents of the binary log file were logged before the current day. For example, if a command line were executed on April 15, 1991, on a binary log file from April 11, no records would be selected, because the file would contain time stamps from only April 11.

The *time* argument to the `-b` and `-e` options is keyed off the function ending times.

The `*** New configuration read in. ***` string is printed whenever the report generator reads in a configuration header record. This string indicates that the `aird`(8) process has reread the AIR configuration file and reinitialized its internal processing work list.

EXAMPLES

The following examples show how you can use the command-line options to look at an `aird`(8) binary log file. All the examples are derived from the same binary log file.

Example 1: The following example shows the default selection of all records and the default information displayed for each record:

```
% airdet -h /usr/spool/air/logs/blog

Product     Function     Type of Message
Name        Name
----------- ------------ ----------------
kernel      response     PROD_AVAILABLE
kernel      response     PROD_AVAILABLE
kernel      response     PROD_AVAILABLE
kernel      response     PROD_AVAILABLE
tcp         smailexist   PROD_AVAILABLE
tcp         ntpdexist    PROD_AVAILABLE
tcp         lpdexist     PROD_AVAILABLE
tcp         namedexist   PROD_AVAILABLE
tcp         netexist     PROD_AVAILABLE
nqs         qfexist      PROD_AVAILABLE
tapes       existence    PROD_AVAILABLE
nqs         existence    PROD_AVAILABLE
msgdaemon   existence    PROD_AVAILABLE
nqs         netexist     PROD_AVAILABLE
tcp         existence    PROD_AVAILABLE
tcp         gatedexist   PROD_UNAVAILABLE
tapes       avrexist     PROD_AVAILABLE
tcp         snmpdexist   PROD_AVAILABLE
kernel      existence    PROD_AVAILABLE
kernel      response     PROD_AVAILABLE
kernel      response     PROD_AVAILABLE
kernel      response     PROD_AVAILABLE
kernel      response     PROD_AVAILABLE
kernel      response     PROD_AVAILABLE
```

Example 2: The following example shows the selection of records pertaining to specific tests of specific products:

```
% airdet -h -p kernel -f response /usr/spool/air/logs/blog

Product     Function     Type of Message
Name        Name
----------- ------------ ----------------
kernel      response     PROD_AVAILABLE
kernel      response     PROD_AVAILABLE
kernel      response     PROD_AVAILABLE
kernel      response     PROD_AVAILABLE
kernel      response     PROD_AVAILABLE
kernel      response     PROD_AVAILABLE
kernel      response     PROD_AVAILABLE
kernel      response     PROD_AVAILABLE
kernel      response     PROD_AVAILABLE
```

Example 3: The following example shows the selection of records with specific return types and the text display option:

```
% airdet -hm -p tcp -T PROD_UNAVAILABLE /usr/spool/air/logs/blog

Product     Function      Type of Message   Message
Name        Name                            Text
----------- ------------- ----------------- ---------------------------
tcp         gatedexist    PROD_UNAVAILABLE  Test Failed
```

Example 4: The following example shows the selection of records whose elapsed time exceeds the specified time and the elapsed time display option:

```
% airdet -hl -p nqs -T PROD_AVAILABLE -n 1 /usr/spool/air/logs/blog

Elapsed    Product     Function      Type of Message
Time       Name        Name
---------- ----------- ------------- ----------------
00:02:10   nqs         netexist      PROD_AVAILABLE
```

Example 5: The following example shows the selection of records whose ending time comes after the specified time and the time-stamp display option:

```
% airdet -ht -b "4/11/91 10:50" -p kernel /usr/spool/air/logs/blog

Start                End                  Product  Function  Type of
Time                 Time                 Name     Name      Message
-------------------- -------------------- -------  --------  ----------
Apr 11 10:50:37 1991 Apr 11 10:50:47 1991 kernel   response  PROD_AVAIL
Apr 11 10:51:37 1991 Apr 11 10:51:47 1991 kernel   response  PROD_AVAIL
Apr 11 10:52:37 1991 Apr 11 10:52:47 1991 kernel   response  PROD_AVAIL
Apr 11 10:53:37 1991 Apr 11 10:53:47 1991 kernel   response  PROD_AVAIL
```

Example 6: The following example shows the selection of records whose ending time comes within the specified range of time and the time-stamp display option:

```
% airdet -ht -b "10:50" -e "10:53" -p kernel /usr/spool/air/logs/blog

Start                End                  Product  Function  Type of
Time                 Time                 Name     Name      Message
-------------------- -------------------- -------- --------  ----------
Apr 11 10:50:37 1991 Apr 11 10:50:47 1991 kernel   response  PROD_AVAIL
Apr 11 10:51:37 1991 Apr 11 10:51:47 1991 kernel   response  PROD_AVAIL
Apr 11 10:52:37 1991 Apr 11 10:52:47 1991 kernel   response  PROD_AVAIL
```

SEE ALSO

aird(8), airprconf(8), airsum(8), airtsum(8)

NAME

`airexist` – Searches the process table in kernel memory for a process matching the command-line requirements

SYNOPSIS

`/usr/air/bin/airexist` [`-a` *acid*] [`-j` *jid*] [`-p` *pid*] [`-u` *uid*] [`-J`] [`-P` *ppid*] [`-e`] [`-v`] [`-c` *corefile*] [`-n` *namelist*] *process_name*

`/usr/air/bin/airexist` [`-a` *acid*] [`-j` *jid*] [`-p` *pid*] [`-u` *uid*] [`-J`] [`-P` *ppid*] [`-I`] [`-v`] [`-c` *corefile*] [`-n` *namelist*] *process_name*

`/usr/air/bin/airexist` [`-a` *acid*] [`-j` *jid*] [`-p` *pid*] [`-u` *uid*] [`-J`] [`-P` *ppid*] [`-M`] [`-v`] [`-c` *corefile*] [`-n` *namelist*] *process_name*

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `airexist` command searches in the kernel memory process table for a process matching the command-line specifications. If options are not specified, `airexist` immediately returns a no match value.

The `airexist` command accepts the following options.

Selection Criteria

`-a` *acid* Makes a match only if *process_name* has account ID (*acid*).

`-j` *jid* Makes a match only if *process_name* has job ID (*jid*).

`-p` *pid* Makes a match only if *process_name* has process ID (*pid*).

`-u` *uid* Makes a match only if *process_name* has user ID (*uid*).

`-J` Makes a match only if *process_name* is in its own job.

`-P` *ppid* Makes a match only if *process_name* has parent process ID (*ppid*).

Exit Status Criteria

(Note that the `-e`, `-I`, and `-M` options are mutually exclusive.)

`-e` Uses `PASSED` and `FAILED` environment variables for status.

`-I` Sets the exit status to the number of matches found.

`-M` If a match is not made, sets exit status to `0`. If one match is made, sets exit status to `1`. If more than one match is made, sets exit status to `2`.

`-v` Prints an ASCII result string to standard output.

Searching Files

`-c` *corefile* Uses the *corefile* file in place of `/dev/mem`.

`-n` *namelist* Takes *namelist* as the name of an alternative system *namelist* file in place of `/unicos`.

process_name Name of the process to match.

FILES

`/unicos`	System namelist
`/dev/mem`	Memory
`/etc/passwd`	Supplies UID information

SEE ALSO

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

NAME

airping – Measures round-trip delays and packet loss across network paths, using the specified protocol

SYNOPSIS

/usr/air/bin/airping [-d] [-i] [-r] [-t] [-u] [-v] [-l *length*] [-n *num*] *host* [*length* [*num*]]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The airping command sends data across the specified protocol and determines the state of the network connection based upon the packet loss value. Available protocols are the Internet Control Message Protocol (ICMP), the Internet Transmission Control Protocol (TCP), and the Internet User Datagram Protocol (UDP).

The airping command accepts the following options and operands:

-d Enables the recording of socket debugging information.

-i Specifies the use of ICMP protocol; this is the default protocol.

-r Enables routing bypass for outgoing messages.

-t Specifies the use of TCP protocol.

-u Specifies the use of UDP protocol.

-v Sets verbose mode.

-l *length* Sets the data length for the packets.

-n *num* Sets the number of packets to send. The default is infinite.

host [*length* [*num*]]
Specifies the name of the host to be used as the end network node, the packet data length, and the number of packets to send.

SEE ALSO

ping(8)

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

NAME

airprconf – Prints AIR configuration file contents from configuration headers in the aird(8) binary log file

SYNOPSIS

airprconf [-a] [-D *number*] [-b *time*] [-e *time*] [-P] *file1* [*file2* ...]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The airprconf command prints the contents of the automated incident reporting (AIR) configuration file read in by the aird(8) daemon at the initiation of its processing or on reception of the SIGHUP signal. The aird(8) daemon logs this information as a configuration header in its binary log file after it reads and translates the contents of the configuration file.

When the -a option is specified, each of the configuration headers in the specified binary log files is processed, and the subsequent configuration file contents are printed. By default, airprconf prints only the last configuration header record in the specified binary log file.

The airprconf command accepts the following options:

-a Prints all configuration headers found. Overrides the default, which is to print only the last header, if any currently exist in the file.

-D *number* Sets the debugging level. The *number* value is the number of debugging messages to print; possible values are integers in the range 1 through 20. A value of 0 sets debugging to off. As the number increases, so does the number of messages.

-b *time* Prints all configuration headers found after the specified *time*. Time format is as follows:

"[*month/day*[/*year*]] *hour:min*[:*sec*]"

-e *time* Prints all configuration headers found before the specified *time*. By default, only the last configuration header is printed. Time format is as follows:

"[*month/day*[/*year*]] *hour:min*[:*sec*]"

-P Prints the configuration data in easy-to-read format as it appears in the configuration file.

file1 [*file2* ...] aird binary log files to use as input. Separate multiple file names, if any, with spaces.

EXAMPLES

Example 1: The following example prints version information and simple configuration information in the default format:

```
% airprconf /usr/spool/air/logs/blog
Type count     2
Message count  4
Product count  2
Function count 5
TYPE 0 Tag: FAILED
TYPE 1 Tag: PASSED
PRODUCT: tcp
Message 0 Prod: tcp Tag: PASSED Text: Test Passed
Message 1 Prod: tcp Tag: FAILED Text: Test Failed
FUNCTION: existence
FUNCTION: function
PRODUCT: tapes
Message 2 Prod: tapes Tag: FAILED Text: Test Failed
Message 3 Prod: tapes Tag: PASSED Text: Test Passed
FUNCTION: existence
FUNCTION: response
FUNCTION: function
```

Example 2: The configuration information printed in the following example is the same as that printed in the previous example, but the format is like that of the configuration file:

```
%  airprconf -P /usr/spool/air/logs/blog
#
# Start of Configuration File Generated by airprconf on Tue Apr 16
# 10:16:02 1991
#
CONFIG  kernel_test_version
        #
        #       Define Coordinator logfile
        #
        COORD_LOG       /usr/spool/air/logs/coord.log
        #
        #       Define Coordinator Heart Beat
        #
        COORD_HBEAT     10
        #
        #       Define Coordinator DEBUG Level (-D)
        #
        COORD_DEBUG     0
        #
        #       Define Coordinator Test Initiation directory (-h)
        #
        COORD_TESTDIR   /usr/air/test
        #
        #       Define Coordinator Binary output file (-f)
        #
        COORD_BLOG      /usr/spool/air/logs/blog
        #
        #       Define Coordinator ASCII Logging Level (-L)
        #
        COORD_LOGLEV    0
        #
        #       Define TYPES
        #
        TYPE    FAILED  PROD_UNAVAILABLE
        TYPE    PASSED  PROD_AVAILABLE
        #
        #       Define product tcp
        #
        PRODUCT tcp     ON
                MESSAGE PASSED  Test Passed
                MESSAGE FAILED  Test Failed
                #
                #       Define Function existence of Product tcp
                #
```

```
        FUNCTION        existence      ON
                RATE    5
                EXECUTE /usr/air/test/tcp/tcp.exist
                LOGFILE NONE
                TIMEOUT NONE
                RETURN  PASSED  0       NONE
                RETURN  FAILED  1       NONE
        ENDFUNCTION     existence
        #
        #       Define Function function of Product tcp
        #
        FUNCTION        function      ON
                RATE    10
                EXECUTE /usr/air/test/tcp/tcp.funct
                LOGFILE NONE
                TIMEOUT NONE
                RETURN  FAILED  1       NONE
                RETURN  PASSED  0       NONE
        ENDFUNCTION     function
ENDPRODUCT      tcp
#
#       Define product tapes
#
PRODUCT tapes   ON
        MESSAGE FAILED  Test Failed
        MESSAGE PASSED  Test Passed
        #
        #       Define Function existence of Product tapes
        #
        FUNCTION        existence      ON
                RATE    5
                EXECUTE /usr/air/test/tapes/tape.exist
                LOGFILE NONE
                TIMEOUT NONE
                RETURN  PASSED  0       NONE
                RETURN  FAILED  1       NONE
        ENDFUNCTION     existence
        #
        #       Define Function response of Product tapes
        #
        FUNCTION        response      ON
                RATE    5
                EXECUTE /usr/air/test/tapes/tape.exist
                LOGFILE NONE
```

```
                        TIMEOUT NONE
                        RETURN  PASSED  0       NONE
                        RETURN  FAILED  1       NONE
                ENDFUNCTION     response
                #
                #       Define Function function of Product tapes
                #
                FUNCTION        function      ON
                        RATE    10
                        EXECUTE /usr/air/test/tapes/tape.funct
                        LOGFILE NONE
                        TIMEOUT NONE
                        RETURN  FAILED  1       NONE
                        RETURN  PASSED  0       NONE
                ENDFUNCTION     function
        ENDPRODUCT      tapes
ENDCONFIG       kernel_test_version
#
# End of Configuration File Generated by airprconf on Tue Apr 16
# 10:16:02 1991
#
```

SEE ALSO

airckconf(8), aird(8), airdet(8), airsum(8), airtsum(8)

NAME

airrep – Produces AIR activity reports

SYNOPSIS

/usr/air/bin/airrep [-o *file*] *logfiles*

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The airrep shell script produces generic activity reports for the automated incident reporting (AIR) product set by invoking the airsum(8), airtsum(8), and airdet(8) report generator commands. The time period over which airrep reports is specified in the binary log file arguments.

By default, airrep invokes the following report generator command lines:

```
airsum -hBA logfiles
airtsum -rhaS logfiles
airdet -T PROD_UNAVAILABLE -hmt logfiles
```

The airsum(8) shell script prints the product availability summary and breakdown information, airtsum(8) prints the monitoring function's information, including the return type breakdowns, and airdet(8) prints the records indicating an unavailable product in the monitored product set. For further information on available options, see the man pages for each report generator.

The airrep shell script accepts the following option and operand:

-o *file* Redirects standard output to the specified file.

logfiles aird(8) binary log files to be used as input.

SEE ALSO

aird(8), airdet(8), airsum(8), airtsum(8)

UNICOS Resource Administration, Cray Research publication SG–2302

NAME

airsum – Generates availability summary reports based on aird(8) binary log file

SYNOPSIS

airsum [-b *time*] [-e *time*] [-a [*key*]] [-u [*key*]] [-h] [-E] [-B] [-l] [-m] [-s] [-t] [-A] [-D *number*] [-L] [-M] [-S] [-T] *file1* [*file2* ...]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The airsum command collects data from the automated incident reporting (AIR) daemon's (aird(8)) binary log file and prints summary statistics on the availability of the monitored products.

Three main forms of statistical presentation are available:

- Periodic breakdown of availability
- Statistical summary of the periodic breakdowns
- Overall availability summary

The final pass at data collection results in a series of availability structures for each product. Each element in this chain indicates the state of the product and the time during which the product was in that state. From this chain, the periodic breakdown summaries and statistics are generated.

The product's state is determined by the results of all of the configured functions; for a product to be determined available, all configured functions must return an available status.

By default, this command prints the names of the monitored products, the total time during which each product was available and unavailable, and the relative and real percentages of time available and unavailable.

The relative percentage column presents the percentage of time that the aird(8) process was running during which each of the products was available. The real percentage column represents the percentage of time each of the products was available during the entire specified interval.

The airsum command accepts the following selection and printing options.

Selection Options

-b *time* Specifies the sample start time. Time format is as follows:

"[*month/day*[*/year*]] *hour:min*[:*sec*]"

-e *time* Specifies the sample end time. Time format is as follows:

"[*month/day*[*/year*]] *hour:min*[:*sec*]"

`-a` [*key*] Specifies the availability of the type key. By default, this key is defined as the `PROD_AVAILABLE` type. Use this option to key a different function return type for the availability determinations.

`-u` [*key*] Specifies the unavailability of the type key. By default, this key is defined as the `PROD_UNAVAILABLE` type. Use this option to key a different function return type for the availability determinations.

Printing Options

`-h` Prints headers.

`-E` Prints a summary of each configuration read. The default is to print only the total summary.

`-B` Prints the elements of the availability breakdown chain. Each element in the chain indicates either a state change, from available to unavailable or vice versa, or a new configuration.

`-l` Prints the longest time period available.

`-m` Prints the average time period available.

`-s` Prints the shortest time period available.

`-t` Prints the total time available.

`-A` Prints the breakdown by period summary report (same as specifying the options `-lmstLMST`).

`-D` *number* Sets the debugging level. The *number* value is the number of debugging messages to print; possible values are integers in the range 1 through 20. A value of `0` sets debugging to off. As the number increases, so does the number of messages.

`-L` Prints the longest time period unavailable.

`-M` Prints the average time period unavailable.

`-S` Prints the shortest time period unavailable.

`-T` Prints the total time unavailable.

file1 [*file2* ...] `aird`(8) binary log files to use as input. One input file is required. You can specify others, separating the file names with spaces.

NOTES

If the date is not included in the time specifications for the `-b` and `-e` options, `airsum` uses today's date. Be careful when using the `-b` and `-e` options without date specifications if you are looking at binary log files no longer directly logged to by `aird`(8). It is possible that no records will be selected because the contents of the binary log file were logged before the current day. For example, if a command line were executed on April 15, 1991, on a binary log file from April 11, no records would be selected because the file would contain time stamps from only April 11.

The *time* argument to the `-b` and `-e` options is keyed off the function ending times.

Multiple log file arguments' contents must be ordered exclusively by time.

EXAMPLES

Example 1: The following example shows the default availability summary information displayed:

```
% airsum -h /usr/spool/air/logs/blog

*** Total Availability Summary ***

Summary Information

Product    Total Time     Total Time     Rel. Perc. Real Perc.
Name       Available      Unavailable    Available  Available
---------- -------------- -------------- ---------- ----------
airdaemon      6:14:41:39       10:20:00        100         93
kernel         6:14:40:30       00:00:00         99         93
tapes          6:06:04:03       06:46:40         94         88
tcp              00:00:00     6:12:50:22          0          0
nqs              07:38:02     6:05:07:19          4          4
-------------------------------------------------------------
```

Example 2: The following example shows an availability summary for each configuration rather than the default action of only the final and grand total summary:

```
% airsum -hE /usr/spool/air/logs/blog

Summary Information

Product    Total Time     Total Time     Rel. Perc. Real Perc.
Name       Available      Unavailable    Available  Available
---------- -------------- -------------- ---------- ----------
airdaemon        02:28:38       00:00:00        100        100
kernel           02:28:38       00:00:00        100        100
tapes            02:27:28       00:00:00         99         99
tcp              00:00:00       02:27:28          0          0
nqs              02:27:28       00:00:00         99         99
-------------------------------------------------------------
```

```
*** New configuration read in. ***

Summary Information

Product    Total Time     Total Time     Rel. Perc. Real Perc.
Name       Available      Unavailable    Available  Available
---------- -------------- -------------- ---------- ----------
airdaemon        00:08:10       00:00:00        100        100
kernel           00:08:10       00:00:00        100        100
tcp              00:00:00       00:04:00          0          0
nqs              00:04:00       00:00:00         48         48
tapes            00:04:00       00:00:00         48         48
-------------------------------------------------------------

*** New configuration read in. ***
*** Total Availability Summary ***

Summary Information

Product    Total Time     Total Time     Rel. Perc. Real Perc.
Name       Available      Unavailable    Available  Available
---------- -------------- -------------- ---------- ----------
airdaemon      6:14:41:39       10:20:00        100         93
kernel         6:14:40:30       00:00:00         99         93
tapes          6:06:04:03       06:46:40         94         88
tcp              00:00:00     6:12:50:22          0          0
nqs              07:38:02     6:05:07:19          4          4
-------------------------------------------------------------
```

Example 3: The following example shows the additional periodic breakdown of product availability that can be requested:

```
%  airsum -hB /usr/spool/air/logs/blog

*** Total Availability Summary ***


Product Availability Breakdown

Product    Product          From                 Until
Name       Status           Time                 Time
---------- ---------------- -------------------- --------------------
airdaemon  PROD_AVAILABLE   Apr 11 08:15:09 1991 Apr 11 10:43:47 1991
           PROD_AVAILABLE   Apr 11 10:45:37 1991 Apr 11 10:53:47 1991
           PROD_AVAILABLE   Apr 11 10:55:37 1991 Apr 11 11:03:47 1991
kernel     PROD_AVAILABLE   Apr 11 08:15:09 1991 Apr 11 10:43:47 1991
           PROD_AVAILABLE   Apr 11 10:45:37 1991 Apr 11 10:53:47 1991
           PROD_AVAILABLE   Apr 11 10:55:37 1991 Apr 11 11:03:47 1991
tapes      PROD_AVAILABLE   Apr 11 08:15:09 1991 Apr 11 10:42:37 1991
           PROD_AVAILABLE   Apr 11 10:45:37 1991 Apr 11 10:49:37 1991
           PROD_AVAILABLE   Apr 11 10:55:37 1991 Apr 11 10:59:37 1991
           PROD_UNAVAILABLE Apr 11 23:54:19 1991 Apr 12 05:59:19 1991
           PROD_AVAILABLE   Apr 12 07:41:08 1991 Apr 12 08:44:01 1991
           PROD_AVAILABLE   Apr 12 09:19:34 1991 Apr 12 16:58:28 1991
           PROD_UNAVAILABLE Apr 16 07:07:20 1991 Apr 16 07:12:20 1991
           PROD_AVAILABLE   Apr 16 07:12:20 1991 Apr 16 07:17:20 1991
           PROD_AVAILABLE   Apr 16 07:17:20 1991 Apr 16 07:37:20 1991
           PROD_UNAVAILABLE Apr 16 07:37:20 1991 Apr 16 07:47:20 1991
           PROD_AVAILABLE   Apr 16 07:47:20 1991 Apr 16 08:12:20 1991
           PROD_AVAILABLE   Apr 16 09:05:10 1991 Apr 16 11:07:30 1991
           PROD_UNAVAILABLE Apr 18 07:29:37 1991 Apr 18 07:51:17 1991
           PROD_UNAVAILABLE Apr 18 07:51:17 1991 Apr 18 07:56:17 1991
           PROD_AVAILABLE   Apr 18 07:56:17 1991 Apr 18 08:46:17 1991
           PROD_AVAILABLE   Apr 18 08:58:37 1991 Apr 18 09:23:11 1991
tcp        PROD_UNAVAILABLE Apr 11 08:15:09 1991 Apr 11 10:42:37 1991
           PROD_UNAVAILABLE Apr 11 10:45:37 1991 Apr 11 10:49:37 1991
           PROD_UNAVAILABLE Apr 11 10:55:37 1991 Apr 11 10:59:37 1991
nqs        PROD_AVAILABLE   Apr 11 08:15:09 1991 Apr 11 10:42:37 1991
           PROD_AVAILABLE   Apr 11 10:45:37 1991 Apr 11 10:49:37 1991
           PROD_AVAILABLE   Apr 11 10:55:37 1991 Apr 11 10:59:37 1991
           PROD_UNAVAILABLE Apr 11 13:11:15 1991 Apr 11 14:05:30 1991
           PROD_UNAVAILABLE Apr 11 14:19:41 1991 Apr 12 05:59:19 1991
           PROD_AVAILABLE   Apr 17 09:10:45 1991 Apr 17 13:10:45 1991
           PROD_UNAVAILABLE Apr 17 13:10:45 1991 Apr 17 17:00:45 1991
           PROD_UNAVAILABLE Apr 17 17:11:45 1991 Apr 17 17:56:14 1991
```

```
Summary Information

Product     Total Time      Total Time      Rel. Perc. Real Perc.
Name        Available       Unavailable     Available  Available
----------- --------------- --------------- ---------- ----------
airdaemon       6:14:49:51        10:20:00        100         93
kernel          6:14:48:42        00:00:00         99         93
tapes           6:06:14:03        06:46:40         94         88
tcp               00:00:00     6:13:00:22          0          0
nqs               07:23:28     6:05:31:53          4          4
-------------------------------------------------------------
```

Example 4: The following example shows the availability statistics that can be displayed for each product:

```
% airsum -hA /usr/spool/air/logs/blog

*** Total Availability Summary ***

Summary Information

Product           Total Time      Total Time      Rel. Perc. Real Perc.
Name              Available       Unavailable     Available  Available
----------------- --------------- --------------- ---------- ----------
airdaemon             6:00:11:41        08:15:07        100         94
kernel                6:00:10:32        00:00:00         99         94
tapes                 5:16:15:29        06:20:00         94         89
tcp                     00:00:00     5:22:35:08          0          0
nqs                     07:23:28     5:15:06:39          5          4
-----------------------------------------------------------------------
```

```
Summary Information for Periods

Product    Total Time  Total Time  Shortest Per. Shortest Per.
Name       Available   Unavailable Available     Unavailable
---------- ----------- ----------- ------------- -------------
airdaemon   6:00:11:41    00:00:00      17:21:57      00:00:00
kernel      6:00:10:32    00:00:00      17:21:57      00:00:00
tapes       5:16:15:29    06:20:00      17:20:47      06:05:00
tcp           00:00:00  5:22:35:08      00:00:00      17:20:24
nqs           07:23:28  5:15:06:39      04:00:00      16:21:25

Product    Longest Per.   Longest  Per.  Average Per.   Average Per.
Name       Available      Unavailable    Available      Unavailable
---------- -------------- -------------- -------------- --------------
airdaemon      6:00:11:41       00:00:00     6:00:11:41       00:00:00
kernel         6:00:10:32       00:00:00       17:21:57       00:00:00
tapes          5:16:15:29       06:20:00       17:20:47       06:05:00
tcp              00:00:00     5:22:35:08       00:00:00       17:20:24
nqs              07:23:28     5:15:06:39       04:00:00       16:21:25

*** Total Availability Summary ***

Summary Information

Product    Total Time     Total Time     Rel. Perc. Real Perc.
Name       Available      Unavailable    Available  Available
---------- -------------- -------------- ---------- ----------
airdaemon      6:15:03:06       10:20:00        100         93
kernel         6:15:01:42       00:00:00         99         93
tapes          6:06:29:05       06:46:40         94         88
tcp              00:00:00     6:13:15:24          0          0
nqs              07:23:28     6:05:46:55          4          4
-------------------------------------------------------------
```

```
Summary Information for Periods

Product    Total Time  Total Time  Longest Per.  Longest Per.
Name       Available   Unavailable Available     Unavailable
---------- ----------- ----------- ------------- --------------
airdaemon   6:15:03:06    00:00:00      17:21:57       00:00:00
kernel      6:15:01:42    00:00:00      17:21:57       00:00:00
tapes       6:06:29:05    06:46:40      17:20:47       06:05:00
tcp           00:00:00  6:13:15:24      00:00:00       17:20:24
nqs           07:23:28  6:05:46:55      04:00:00       16:21:25


Product    Shortest Per. Shortest Per.  Average Per.   Average Per.
Name       Available     Unavailable    Available      Unavailable
---------- ------------- -------------- -------------- --------------
airdaemon       00:04:10       00:00:00       03:58:34       00:00:00
kernel          00:04:10       00:00:00       03:58:32       00:00:00
tapes           00:04:00       00:05:00       03:29:58       01:21:20
tcp             00:00:00       00:04:00       00:00:00       03:55:53
nqs             00:04:00       00:00:00       00:27:43       05:09:53
```

Example 5: The following example shows the selection of certain fields for display from the availability statistics for each product in the previous example:

```
% airsum -hlLsS /usr/spool/air/logs/blog

*** Total Availability Summary ***

Summary Information

Product    Total Time     Total Time     Rel. Perc. Real Perc.
Name       Available      Unavailable    Available  Available
---------- -------------- -------------- ---------- ----------
airdaemon      6:15:41:54       10:20:00        100         93
kernel         6:15:40:45       00:00:00         99         93
tapes          6:07:04:03       06:46:40         94         88
tcp              00:00:00     6:13:50:22          0          0
nqs              07:23:28     6:06:21:53          4          4
-------------------------------------------------------------
```

```
Summary Information for Periods

Product    Longest Per.   Longest Per.   Shortest Per.  Shortest Per.
Name       Available      Unavailable    Available      Unavailable
---------- -------------- -------------- -------------- --------------
airdaemon        17:21:57       00:00:00       00:04:10       00:00:00
kernel           17:21:57       00:00:00       00:04:10       00:00:00
tapes            17:20:47       06:05:00       00:04:00       00:05:00
tcp              00:00:00       17:20:24       00:00:00       00:04:00
nqs              04:00:00       16:21:25       00:04:00       00:00:00
```

Example 6: The following example shows the specification of different keys for availability determination (the default is PROD_AVAILABLE and PROD_UNAVAILABLE). In this example, they have been switched.

```
% airsum -h -aPROD_UNAVAILABLE -uPROD_AVAILABLE /usr/spool/air/logs/blog

*** Total Availability Summary ***

Summary Information

Product    Total Time     Total Time     Rel. Perc. Real Perc.
Name       Available      Unavailable    Available  Available
---------- -------------- -------------- ---------- ----------
airdaemon      6:16:00:56       10:20:00        100         93
kernel           00:00:00     6:15:59:47          0          0
tapes            06:25:00     6:07:45:43          4          3
tcp              00:00:00     6:14:10:22          0          0
nqs              11:54:36     6:02:10:45          7          6
-------------------------------------------------------------
```

Example 7: The following example shows the selection of a summary sample time that begins with the specified time:

```
% airsum -h -b "4/17/91 10:00" /usr/spool/air/logs/blog

*** Total Availability Summary ***

Summary Information

Product    Total Time     Total Time     Rel. Perc. Real Perc.
Name       Available      Unavailable    Available  Available
---------- -------------- -------------- ---------- ----------
airdaemon        23:41:18       02:41:15        100         89
tapes            23:14:53       00:20:00         98         88
tcp              00:00:00       23:34:53          0          0
nqs              00:00:00       23:34:53          0          0
kernel           23:45:11       00:00:00        100         90
------------------------------------------------------------
```

Example 8: The following example shows the selection of a summary sample that comes within the specified range of time:

```
% airsum -h -b "9:00" -e "10:00" /usr/spool/air/logs/blog

*** Total Availability Summary ***

Summary Information

Product    Total Time     Total Time     Rel. Perc. Real Perc.
Name       Available      Unavailable    Available  Available
---------- -------------- -------------- ---------- ----------
airdaemon        00:50:50       00:00:01        100         99
kernel           00:50:51       00:00:00        100         99
nqs              00:00:00       00:50:01          0          0
tcp              00:00:00       00:50:01          0          0
tapes            00:50:01       00:00:00         98         98
------------------------------------------------------------
```

SEE ALSO

aird(8), airdet(8), airprconf(8), airtsum(8)

UNICOS Resource Administration, Cray Research publication SG–2302

NAME

airtsum – Generates detailed AIR reports based on aird(8) binary log file

SYNOPSIS

/usr/air/bin/airtsum [-b *time*] [-e *time*] [-h] [-E] [-r] [-S] [-l] [-s] [-a] [-p] [-c] [-D *number*] *file1* [*file2* ...]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The airtsum command gathers data from the event records in the automated incident reporting (AIR) daemon's (aird(8)) binary log file and prints summary statistics on the monitoring functions executed by the AIR daemon. This information is useful when you are analyzing the reports generated by the airsum(8) command. A clearer indication of the testing activity for each product gives credibility to the statistics reported by airsum(8).

This information is also useful in targetting functions that may be incorrectly configured for the system. For example, you may be executing the test for the network daemon component of the Network Queuing System (NQS) while not having that component configured on your system. The function would always return a failed status and, as a result, NQS would always be reported as unavailable. In this case, you could disable that function in the configuration file and obtain more accurate statistics on the availability of NQS.

The default action for this command is to print the name of the functions that were executed for each product, the number of times each function was executed, and the total time the product was tested by each function.

The airtsum command accepts the following options and operands:

-b *time* Specifies the sample start time. Time format is as follows:

"[*month/day*[/*year*]] *hour:min*[:*sec*]"

-e *time* Specifies the sample end time. Time format is as follows:

"[*month/day*[/*year*]] *hour:min*[:*sec*]"

-h Prints headers.

-E Prints a summary of each configuration read. The default is to print only the total summary.

-r Prints a breakdown of return values per function. This information includes a list of return types and the number of times that each type was returned for each function.

-S Prints the beginning and ending times the products were tested by each function.

-l Prints the longest time interval between function executions.

`-s` Prints the shortest time interval between function executions.

`-a` Prints the average time interval between function executions.

`-p` Prints the percentage of time the products were tested by each function.

`-c` Prints the configured time interval between test execution for each function as defined in the configuration header.

`-D` *number* Sets the debugging level. The *number* value is the number of debugging messages to print; possible values are integers in the range 1 through 20. A value of `0` sets debugging to off. As the number increases, so does the number of messages.

file1 [*file2* ...] `aird` binary log files to use as input. One input file is required. You can specify others, separating the file names with spaces.

NOTES

If the date is not included in the time specifications for the `-b` and `-e` options, `airtsum` uses today's date. Be careful when using the `-b` and `-e` options without date specifications if you are looking at binary log files no longer directly logged to by `aird`(8). It is possible that no records will be selected because the contents of the binary log file were logged before the current day. For example, if a command line were executed on April 15, 1991, on a binary log file from April 11, no records would be selected, because the file would contain time stamps from only April 11.

The *time* argument to the `-b` and `-e` options is keyed off the function ending times.

EXAMPLES

Example 1: The following example shows the default testing summary information displayed:

```
% airtsum -h /usr/spool/air/logs/blog

*** Total Test Summary ***

Function Summary Information

Product    Function      Total    Total Time
Name       Name          Executed Tested
---------- ------------- -------- --------------
kernel     response          9312     7:00:37:41
           existence         1835     7:00:30:34
tapes      avrexist          1840     7:00:30:32
           existence         1840     7:00:30:32
           response           906     7:00:25:32
tcp        namedexist        1840     7:00:30:32
           gatedexist        1840     7:00:30:32
           ntpdexist         1840     7:00:30:31
           snmpdexist        1840     7:00:30:32
           smailexist        1840     7:00:30:32
           existence         1840     7:00:30:32
           lpdexist          1840     7:00:30:32
           function           907     7:00:25:31
nqs        qfexist           1840     7:00:30:32
           netexist          1840     7:00:30:31
           existence         1840     7:00:30:32
           response           907     7:00:25:31
           function           600     6:21:32:54
           slexist            295     6:21:17:38
```

Example 2: The following example shows a testing summary for each configuration rather than the default action of the final and grand total summary:

```
% airtsum -hE /usr/spool/air/logs/blog

*** Sample Test Summary ***

Function Summary Information

Product    Function      Total    Total Time
Name       Name          Executed Tested
---------- ------------- -------- --------------
kernel     response            11       00:10:10
           existence            2       00:05:04
tapes      avrexist             2       00:05:00
           existence            2       00:05:00
           response             1       00:00:00
tcp        namedexist           2       00:05:00
           gatedexist           2       00:05:00
           ntpdexist            2       00:05:00
           snmpdexist           2       00:05:00
           smailexist           2       00:05:00
           existence            2       00:05:00
           lpdexist             2       00:05:00
           function             1       00:00:00
nqs        qfexist              2       00:05:00
           netexist             2       00:05:00
           existence            2       00:05:00
           response             1       00:00:00

*** Sample Test Summary ***

Function Summary Information

Product    Function      Total    Total Time
Name       Name          Executed Tested
---------- ------------- -------- --------------
kernel     response             9       00:08:10
           existence            1       00:00:03
tcp        smailexist           1       00:00:00
           ntpdexist            1       00:00:00
           lpdexist             1       00:00:00
           namedexist           1       00:00:00
           existence            1       00:00:00
```

```
            gatedexist           1       00:00:00
            snmpdexist           1       00:00:00
nqs         qfexist              1       00:00:00
            existence            1       00:00:00
            netexist             1       00:00:00
tapes       existence            1       00:00:00
            avrexist             1       00:00:00


*** Sample Test Summary ***
.
.
.

*** Total Test Summary ***


Function Summary Information

Product     Function      Total    Total Time
Name        Name          Executed Tested
----------- ------------- -------- --------------
kernel      response          9317     7:00:42:41
            existence         1836     7:00:35:36
tapes       avrexist          1841     7:00:35:33
            existence         1841     7:00:35:32
            response           906     7:00:25:32
tcp         namedexist        1841     7:00:35:32
            gatedexist        1841     7:00:35:32
            ntpdexist         1841     7:00:35:32
            snmpdexist        1841     7:00:35:32
            smailexist        1841     7:00:35:32
            existence         1841     7:00:35:32
            lpdexist          1841     7:00:35:32
            function           907     7:00:25:31
nqs         qfexist           1841     7:00:35:32
            netexist          1841     7:00:35:33
            existence         1841     7:00:35:32
            response           907     7:00:25:31
            function           600     6:21:32:54
            slexist            295     6:21:17:38
```

Example 3: The following example shows the return type breakdown for each function:

```
% airtsum -hr /usr/spool/air/logs/blog

*** Total Test Summary ***

Function Summary Information

Product    Function    Total    Return          Number   Total Time
Name       Name        Executed Type            Returned Tested
---------- ----------- -------- -------------- -------- --------------
kernel     response        9321 PROD_AVAILABLE     9321     7:00:46:41
           existence       1837 PROD_AVAILABLE     1837     7:00:40:33
tapes      avrexist        1842 PROD_AVAILABLE     1764     7:00:40:31
                                PROD_UNAVAILABLE     78
           existence       1842 PROD_AVAILABLE     1838     7:00:40:31
                                PROD_UNAVAILABLE      4
           response         907 PROD_AVAILABLE      904     7:00:35:31
                                PROD_UNAVAILABLE      3
tcp        namedexist      1842 PROD_AVAILABLE     1842     7:00:40:31
           gatedexist      1842 PROD_UNAVAILABLE   1842     7:00:40:31
           ntpdexist       1842 PROD_AVAILABLE     1842     7:00:40:31
           snmpdexist      1842 PROD_AVAILABLE     1842     7:00:40:31
           smailexist      1842 PROD_AVAILABLE     1842     7:00:40:31
           existence       1842 PROD_AVAILABLE     1842     7:00:40:31
           lpdexist        1842 PROD_AVAILABLE     1842     7:00:40:31
           function         908 PROD_UNAVAILABLE    908     7:00:35:31
nqs        qfexist         1842 PROD_AVAILABLE     1742     7:00:40:31
                                PROD_UNAVAILABLE    100
           netexist        1842 PROD_AVAILABLE     1780     7:00:40:31
                                PROD_UNAVAILABLE     62
           existence       1842 PROD_AVAILABLE     1743     7:00:40:31
                                PROD_UNAVAILABLE     99
           response         908 PROD_AVAILABLE      859     7:00:35:31
                                PROD_UNAVAILABLE     49
           function         600 PROD_UNAVAILABLE    600     6:21:32:54
           slexist          295 PROD_UNAVAILABLE    295     6:21:17:38
```

Example 4: The following example shows the range of times during which the functions are monitoring their products:

```
% airtsum -hS /usr/spool/air/logs/blog

*** Total Test Summary ***

Function Summary Information

Product   Function   Total  Total Time  From             Until
Name      Name       Exec.  Tested      Time             Time
--------- ---------- ------ ----------- ---------------- ---------------
kernel    response     9324  7:00:49:41 Apr 11 10:33:37  Apr 18 11:23:18
          existence    1838  7:00:45:34 Apr 11 10:37:37  Apr 18 11:23:11
tapes     avrexist     1843  7:00:45:31 Apr 11 10:37:37  Apr 18 11:23:08
          existence    1843  7:00:45:31 Apr 11 10:37:37  Apr 18 11:23:08
          response      907  7:00:35:31 Apr 11 10:42:37  Apr 18 11:18:08
tcp       namedexist   1843  7:00:45:31 Apr 11 10:37:37  Apr 18 11:23:08
          gatedexist   1843  7:00:45:31 Apr 11 10:37:37  Apr 18 11:23:08
          ntpdexist    1843  7:00:45:31 Apr 11 10:37:37  Apr 18 11:23:08
          snmpdexist   1843  7:00:45:31 Apr 11 10:37:37  Apr 18 11:23:08
          smailexist   1843  7:00:45:31 Apr 11 10:37:37  Apr 18 11:23:08
          existence    1843  7:00:45:31 Apr 11 10:37:37  Apr 18 11:23:08
          lpdexist     1843  7:00:45:31 Apr 11 10:37:37  Apr 18 11:23:08
          function      908  7:00:35:31 Apr 11 10:42:37  Apr 18 11:18:08
nqs       qfexist      1843  7:00:45:31 Apr 11 10:37:37  Apr 18 11:23:08
          netexist     1843  7:00:45:31 Apr 11 10:37:37  Apr 18 11:23:08
          existence    1843  7:00:45:31 Apr 11 10:37:37  Apr 18 11:23:08
          response      908  7:00:35:31 Apr 11 10:42:37  Apr 18 11:18:08
          function      601  6:21:47:54 Apr 11 13:35:30  Apr 18 11:23:24
          slexist       295  6:21:17:38 Apr 11 13:50:30  Apr 18 11:08:08
```

Example 5: The following example shows some of the interval statistics that can be displayed for each function:

```
% airtsum -hlsc /usr/spool/air/logs/blog

*** Total Test Summary ***

Function Summary Information

Product   Function   Total  Total Time    Long      Short     Configured
Name      Name       Exec.  Tested        Interval  Interval  Interval
--------- ---------- ------ ------------- --------- --------- ----------
kernel    response     9329    7:00:54:41  00:01:54  00:00:01  00:01:00
          existence    1839    7:00:50:33  00:05:20  00:00:11  00:05:00
tapes     avrexist     1844    7:00:50:31  00:05:22  00:04:10  00:05:00
          existence    1844    7:00:50:31  00:05:14  00:04:10  00:05:00
          response      908    7:00:45:31  00:19:55  00:09:10  00:10:00
tcp       namedexist   1844    7:00:50:31  00:05:13  00:03:48  00:05:00
          gatedexist   1844    7:00:50:31  00:05:22  00:04:10  00:05:00
          ntpdexist    1844    7:00:50:31  00:05:22  00:03:48  00:05:00
          snmpdexist   1844    7:00:50:31  00:05:22  00:04:10  00:05:00
          smailexist   1844    7:00:50:31  00:05:14  00:04:10  00:05:00
          existence    1844    7:00:50:31  00:05:22  00:04:10  00:05:00
          lpdexist     1844    7:00:50:31  00:05:22  00:03:48  00:05:00
          function      909    7:00:45:31  00:10:10  00:07:45  00:10:00
nqs       qfexist      1844    7:00:50:31  00:05:22  00:03:48  00:05:00
          netexist     1844    7:00:50:31  00:05:22  00:02:45  00:05:00
          existence    1844    7:00:50:31  00:05:22  00:04:10  00:05:00
          response      909    7:00:45:31  00:10:10  00:09:10  00:10:00
          function      601    6:21:47:54  00:15:00  00:13:49  00:15:00
          slexist       295    6:21:17:38  00:30:01  00:29:10  00:30:00
```

Example 6: The following example shows more of the interval statistics that can be displayed for each function:

```
% airtsum -hap /usr/spool/air/logs/blog

*** Total Test Summary ***

Function Summary Information

Product    Function      Total    Total Time     Percent  Average
Name       Name          Executed Tested         Time     Interval
---------- ------------- -------- -------------- -------- -------------
kernel     response          9343     7:01:08:42       98      00:00:49
           existence         1841     7:01:00:35       98      00:04:50
tapes      avrexist          1846     7:01:00:31       98      00:04:53
           existence         1846     7:01:00:31       98      00:04:53
           response           909     7:00:55:31       98      00:09:43
tcp        namedexist        1846     7:01:00:31       98      00:04:53
           gatedexist        1846     7:01:00:31       98      00:04:53
           ntpdexist         1846     7:01:00:31       98      00:04:53
           snmpdexist        1846     7:01:00:31       98      00:04:53
           smailexist        1846     7:01:00:31       98      00:04:53
           existence         1846     7:01:00:31       98      00:04:53
           lpdexist          1846     7:01:00:31       98      00:04:53
           function           910     7:00:55:31       98      00:09:42
nqs        qfexist           1846     7:01:00:31       98      00:04:53
           netexist          1846     7:01:00:31       98      00:04:53
           existence         1846     7:01:00:31       98      00:04:53
           response           910     7:00:55:31       98      00:09:42
           function           602     6:22:02:53       96      00:14:08
           slexist            296     6:21:47:39       96      00:27:27
```

Example 7: The following example shows the selection of a summary sample time that begins with the specified time:

```
% airtsum -h -b "4/17/91 10:00" /usr/spool/air/logs/blog

*** Total Test Summary ***

Function Summary Information

Product          Function         Total    Total Time
Name             Name             Executed Tested
---------------- ---------------- -------- --------------
tapes            existence             281     1:02:07:23
                 avrexist              281     1:02:07:24
                 response              140     1:02:02:23
tcp              snmpdexist            281     1:02:07:23
                 namedexist            281     1:02:07:23
                 smailexist            281     1:02:07:23
                 existence             281     1:02:07:23
                 lpdexist              281     1:02:07:23
                 ntpdexist             281     1:02:07:23
                 gatedexist            281     1:02:07:23
                 function              140     1:02:02:23
nqs              existence             281     1:02:07:24
                 netexist              281     1:02:07:23
                 qfexist               281     1:02:07:23
                 response              140     1:02:02:23
                 function               92     1:01:57:39
                 slexist                46     1:01:42:23
kernel           existence             280     1:02:07:26
                 response             1420     1:02:10:32
```

Example 8: The following example shows the selection of a summary sample that comes within the specified range of time:

```
% airtsum -h -b "9:00" -e "10:00" /usr/spool/air/logs/blog

*** Total Test Summary ***

Function Summary Information

Product          Function         Total    Total Time
Name             Name             Executed Tested
---------------- ---------------- -------- --------------
kernel           response               52       00:50:50
                 existence              10       00:45:03
nqs              netexist               10       00:45:01
                 qfexist                10       00:45:01
                 existence              10       00:45:01
                 response                5       00:40:01
                 function                3       00:30:16
                 slexist                 1       00:00:00
msgdaemon        existence              10       00:45:01
                 response                5       00:40:01
tcp              existence              10       00:45:01
                 lpdexist               10       00:45:01
                 ntpdexist              10       00:45:01
                 snmpdexist             10       00:45:01
                 namedexist             10       00:45:01
                 gatedexist             10       00:45:01
                 smailexist             10       00:45:01
                 function                5       00:40:01
nqs-superlink    slexist                10       00:45:01
tapes            avrexist               10       00:45:01
                 existence              10       00:45:01
                 response                5       00:40:01
```

SEE ALSO

aird(8), airdet(8), airprconf(8), airsum(8)

NAME

arp – Displays address resolution display and control

SYNOPSIS

```
arp hostname
arp -a
arp -d hostname
arp -s hostname ether_addr [temp] [pub]
arp -f filename
```

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The arp program displays and modifies the Internet-to-Ethernet address translation tables that the address resolution protocol (ARP) uses. See the arp(4P) man page for a description of the translation tables.

With no flags specified, the program displays the current ARP entry for *hostname*. You can verify the host by name or by number, using Internet dot notation.

The arp command accepts the following options:

`-a` Displays the current ARP entries by reading the table via sysctl(3C).

`-d` *hostname* Deletes an entry for the host called *hostname*.

`-s` *hostname ether_addr* [temp] [pub]
Creates an ARP entry for the host called *hostname* with the Ethernet address *ether_addr*. The Ethernet address is specified as 6 hexadecimal bytes separated by colons. The entry is permanent unless you specify the temp argument in the command. If you specify the pub argument, the entry will be published (for example, this system acts as an ARP server, responding to requests for *hostname* even though the host address is not its own).

`-f` *filename* Causes the file named *filename* to be read and multiple entries to be set in the ARP tables. Entries in the file should have the following form:

hostname ether_addr [temp] [pub]

hostname Displays the current ARP entry for the specified host name or number.

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

`inet`(3C) to manipulate internet addresses

`ifconfig`(8) to configure network interface parameters

`privtext`(1) for information about getting the privilege text of a file in the *UNICOS User Commands Reference Manual*, Cray Research publication SR–2011

`arp`(4P) for information on the address resolution protocol 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

arrayd – Array services daemon

SYNOPSIS

arrayd [-c] [-f *filename* ...] [-m *number*] [-n] [-nf] [-nm] [-p *number*] [-qf] [-qm] [-sm] [-v...]

IMPLEMENTATION

IRIX and UNICOS systems

DESCRIPTION

The arrayd command invokes the array services daemon. The daemon performs several different tasks related to the use of an array of two or more machines, including the following:

- Allocating global array session handles
- Forwarding array commands to all of the machines in an array
- Maintaining a database of the current array configuration and providing that information to other commands and programs
- Determining which processes belong to a particular array session and providing that information to other commands and programs

The arrayd command itself has several command line options. The valid options include the following:

-c
: Causes arrayd to read any configuration files and then exit immediately, sending any errors to stderr rather than syslog (which is the usual behavior). This is primarily of use for checking the validity of new configuration files. This is the same as the -qf option.

-f *filename*
: Specifies the name of a single configuration file. This option may be specified more than once, in which case the files will be processed in the order in which they are specified. One reason to have multiple configuration files is to allow all of the machines in an array to use a single file for array entries (accessed, for example, through the network file system (NFS)) and still maintain private configuration files for local options and/or security information. The format of an arrayd configuration file is described in arrayd.conf(5). If no configuration files are specified, then /usr/lib/array/arrayd.conf and /usr/lib/array/arrayd.auth will both be used (in that order).

-m *number*
: Sets the machine identifier used by the array services daemon for generating global array session handles to *number*. UNICOS may also use this value when generating array session handles. *number* must be a value between 0 and 32767. It will override any IDENT setting in the LOCAL section of any configuration file.

`-n` Ordinarily, `arrayd` will automatically *daemonize* itself; that is, it will disassociate itself from the current terminal and place itself in the background. Specifying this option causes `arrayd` to run in the foreground on the current terminal. This is mostly useful for testing purposes.

`-nf` Specifies that no configuration files are to be read. This is most useful with options like `-sm` that cause `arrayd` to quit after performing tasks that do not require configuration information. `-nf` will override any `-f` options.

`-nm` Specifies that the system machine ID should not be set. This is used to override a `LOCAL OPTIONS SETMACHID` statement in the configuration file.

`-p` *number* Specifies the port on which the array services daemon should listen for requests. It will override any `PORT` setting in the `LOCAL` section of any configuration file.

`-qf` Directs `arrayd` to quit after parsing the configuration file(s). This is the same as the `-c` option.

`-qm` Directs `arrayd` to quit after setting the system machine ID. This causes `arrayd` to exit as soon as it has set the system machine identifier. This may be useful in cases where a nondefault system machine identifier is desired, but none of the other array services provided by `arrayd` are needed. This can also be used to change the machine identifier on a system that is already running another copy of `arrayd`; in this case, kernel-generated array session handles will use the new machine identifier, while those generated by `arrayd` will continue to use the original machine identifier.

`-sm` Some versions of UNICOS permit setting a system machine identifier, which is used by the kernel for generating global array session handles. If the current system has this facility, and `-sm` is specified, `arrayd` will set the machine ID to the value specified by a `LOCAL IDENT` statement in the configuration file or on the command line by using the `-m` option.

`-v` Specifies the verbose mode: the daemon runs in the foreground (as with the `-n` option) and sends any error messages, plus some additional messages, to `stderr` rather than `syslog`. Specifying this option more than once, or specifying more than one `v` (for example `vvv`), causes additional debugging information to be generated.

NOTES

The `arrayd` command can be set up to run automatically at system initialization time by editing `/etc/config/daemons` to turn on the array feature. To do this, modify the `arrayd` line to read `YES` instead of `NO`.

WARNINGS

An `/etc/hosts.equiv` file is created when the UNICOS system is built; a `localhost` line, which enables array services and the message passing interface (MPI) to run, is automatically put in this file. If you have an existing `/etc/hosts.equiv` file on your system, you need to manually add the `localhost` line to your file.

SEE ALSO

`arrayd.conf`(5)

`array_services`(7), `array_sessions`(7)

NAME

atmadmin – Configures and displays ATM administration statistics

SYNOPSIS

/etc/atmadmin [-u *unit_number*] [-aCdhqsSt] [-c *param*] [-i *VCI*] [-I *VCI*]

IMPLEMENTATION

Cray Research systems with IOS model E and HIPPI channel

CRAY J90 series

CRAY EL series

Note: This command is available only for systems that do not have GigaRing technology.

DESCRIPTION

The atmadmin command configures and displays statistics for the Cray Research Asynchronous Transfer Mode (ATM) driver.

The atmadmin command accepts the following options:

-u *unit_number* Specifies the unit number to use for issuing the command. For example, you would use 0 for interface atm0.

-a Displays the current settings on the OC3 SONET user network interface (SUNI) chip. This command does not function properly with the transparent asynchronous transmitter/receiver interface (TAXI) ATM cards.

-C Clears the IOS trace buffer.

-d Toggles IOS tracing on or off.

-h Displays the hardware heartbeat and system flags. This can be used to determine if the hardware is still functioning.

-q Displays IOS queue information.

-s Displays the statistics kept on the IOS.

-S Displays the statistics kept on the ATM hardware.

-t Dumps an IOS trace.

-c *param* Changes the OC3 configuration on the ATM card. This command does not function properly with the TAXI ATM cards. *param* can take one of the following forms:

- Provides register, value, mask values in the form register:value:mask. For example, 3A:02:FF sets register 0x3a to value 0x02 with a setting mask of 0xff.
- Sets the registers by using one of the following key words:

`sonet`	Sets up SUNI chip to run in synchronous optical network (SONET) mode
`sdh`	Sets up SUNI chip to run in synchronous digital hierarchy (SDH) mode
`internal`	Uses internal timing for cell transmission
`external`	Uses external timing for cell transmission
`loopback`	Enables internal loopback
`wire`	Disables internal loopback
`fscramble`	Enables frame scrambling
`nfscramble`	Disables frame scrambling
`sscramble`	Enables stream scrambling
`nsscramble`	Disables stream scrambling
`idle`	Uses idle cell insertion
`unassigned`	Uses unassigned cell insertion

`-i` *VCI* Activates an input Virtual Channel Identifier (VCI) on the ATM card. Usually, all of the input VCIs are activated when the device is configured up. Not recommended for normal use.

`-I` *VCI* Deactivates an input VCI on the ATM card. Not recommended for normal use.

SEE ALSO

`atmarp`(8)

NAME

atmarp – Configures and displays an ATM ARP table

SYNOPSIS

/etc/atmarp -a [-v]

/etc/atmarp -s *host ifc aal vpi vci* [*qos*]

/etc/atmarp -d *host*

IMPLEMENTATION

Cray Research systems with IOS model E and HIPPI channel

CRAY J90 series

CRAY EL series

DESCRIPTION

The atmarp command configures and displays the Asynchronous Transfer Mode (ATM) address resolution protocol (ARP) table on Cray Research systems that have a Native ATM connection or BBG ATM connection.

The atmarp command accepts the following options:

-a Displays the current ATM ARP table entries.

-v Produces a verbose display of the table entries.

-s Sets a permanent ATM ARP table entry. The following are the parameters:

host The IP host name of the remote host. If it is an alias, it must be in the /etc/hosts file.

ifc Interface name (for example, atm0, atm1, bbg0:atm1 ...) that this system uses to reach the remote host. The name is as it appears in the output of the netstat(1B) command with the -i option.

aal The ATM adaption layer (AAL) to be used by this Permanent Virtual Circuit (PVC). This value is based on ATM standards. Specify this number in decimal form.

vpi The Virtual Path Identifier (VPI). The VPI is placed into each ATM cell header so that the cell can be routed through the ATM network. Specify this number in decimal form.

vci The Virtual Channel Identifier (VCI). The VCI is placed into each ATM cell header so that the cell can be routed through the ATM network. This number should be between 32 and 1023. Consult your local network administrator when determining the VCI. Specify this number in decimal form.

qos The quality of service. This is the peak data rate (expressed in kilobits per second), at which this host delivers ATM cells to the remote host through the ATM interface. The default value of 0 causes the peak rate control feature to be disabled when sending to this remote host, thus allowing unlimited bandwidth. Specify this number in decimal form.

`-d` *host* Deletes an ATM ARP table entry associated with the host specified.

SEE ALSO

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

`bbg`(4) (Available only online)

NAME

automount – Mounts NFS or NFS version 3 file systems automatically

SYNOPSIS

/etc/automount [-f *mapfile*] [-m] [-n] [-tl *duration*] [-tm *interval*] [-tw *interval*] [-v] [-M *tmp_dir*] [-T] [*directory* [*mapname*] [-*mount-options*]]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The automount daemon automatically and transparently mounts an network file system (NFS) whenever a file or directory within that system is opened. automount creates a daemon (using fork(2), which appears to the kernel to be an NFS server; this daemon intercepts lookups on the specified *directory* and uses the map contained in *mapname* to determine the server, exported file system, and appropriate mount options for a given file system. The specified map must be a file on the local system. *directory* is a full path name starting with a /.

When supplied, -*mount-options* consists of a leading - and a comma-separated list of mount(8) options; if mount options are specified in the map, however, those in the map take precedence.

When the file system is mounted, members of the *directory* are made available by use of a symbolic link to the real mount point within a temporary directory. If *directory* does not exist, the daemon creates it; it is removed automatically when the daemon exits.

The automount daemon accepts the following options:

-f *mapfile*	Specifies the automount map file.
-m	Suppresses the initialization of *directory-mapname* pairs listed in the auto.master network information service (NIS) database.
-n	Disables dynamic mounts. With this option, references made through the automount daemon succeed only when the target file system was mounted previously. This option can be used to prevent NFS servers from cross-mounting each other.
-tl *duration*	Specifies the number of seconds that a looked-up name remains cached when not in use. The default is 5 minutes.
-tm *interval*	Specifies the number of seconds between attempts to mount a file system. The default is 30 seconds.
-tw *interval*	Specifies the number of seconds between attempts to dismount file systems that have exceeded their cached times. The default is 60 seconds.
-v	Specifies verbose mode. automount reports what it is doing on standard output.
-M *tmp_dir*	Specifies the directory under which the real NFS mounts occur.

`-T`	Specifies trace mode. Expands each NFS call. You must redirect the output of this option to a file.
directory	Specifies the name of the directory on which lookups are performed.
mapname	Specifies the name of the map that contains the mount options.
-mount-options	Specifies a list of mount options.

Maps

An `automount` map is composed of a list of mappings, with one mapping per line. Each mapping is composed of the following fields:

basename [*-mount-options*] *location* [...]

The *basename* field is the name of a subdirectory within the *directory* specified in the `automount` command line or in a master map specified by the `-f` option; it is not a relative path name. The *location* field consists of an entry of the following form:

host `:`*directory*[`:`*subdir*]

The *host* field is the name of the host from which to mount the file system, *directory* is the path name of the directory to mount, and *subdir* (when supplied) is the name of a subdirectory to which the symbolic link is made. You can use this argument to prevent duplicate mounts when multiple directories in the same remote file system are accessed.

You can continue a mapping across line breaks by using a \ as the last character before the new-line character. Comments begin with # and end at the subsequent new-line character.

If more than one *location* is supplied, there is no guarantee as to which location will be used; the first location to respond to the mount request gets mounted. The *mount-options* field can be used to supply options to the `mount`(8) command for the mounted file system.

Since the automounter maps do not have the ability to specify a file system type, specifying '`NFS3`' on either the '*mount-options*' command line or in an automounter map file will cause the automounter to mount a NFS version 3 file system type instead of an NFS file system type. If your system is not licensed for ONC+™ this *mount-option* is ignored.

Special Maps

Currently, two special maps are available. The `-hosts` map specifies mounts of all exported file systems from any host. For example, if the following `automount` command is already in effect, a reference to `/net/hermes/usr` will initiate an automatic mount of all file systems from `hermes` that `automount` can mount:

```
automount -m /net -hosts
```

References to a directory under `/net/hermes` refer to the corresponding directory on `hermes`. The `-passwd` map uses the `passwd`(5) database in its attempt to locate the home directory of a user. For example, assume that the following `automount` command is already in effect:

```
automount -m /homes -passwd
```

If the home directory shown in the password entry for the user *username* has the form */dir/server/username*, and *server* matches the host system on which that directory resides, the result of references to files in `/homes/`*username* is that the file system containing that directory is mounted if necessary, and all such references refer to that user's home directory.

BUGS

Shell file name expansion does not apply to objects not currently mounted or cached. For example, in the preceding example, the `ls /net/*` command might not list `hermes` as a subdirectory of `/net`.

EXAMPLES

The following example provides `automount` access to the exported file systems of any host in the `/etc/hosts` file, by prefixing the path name with `/net/`*hostname*`/ :`, as follows:

```
tutorial# automount  -m  /net  -hosts
```

You can then perform any directory operation on the `/net` subdirectory, as in the following example, using `ls`(1):

```
tutorial% ls /net/hermes/usr/src
```

FILES

`/tmp_mnt`	Directory under which file systems are mounted dynamically by default; you can specify this directory by using the `-M` option.

SEE ALSO

`mount`(8)

NAME

bb – Creates relative bad block file from ASCII flaw table files

SYNOPSIS

/etc/bb [-g] [-p] *special* [*aft_files*]

/etc/bb [-g] [-p] -D *device* -C *cylinder* -H *head* -S *sector special*

IMPLEMENTATION

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

DESCRIPTION

The bb command has two forms. The first format takes input from one or more ASCII flaw tables (aft(5)) files and writes to the standard output a relative bad block table for the given special file, *special*. By default, bb references files in the /etc/aft directory. These files are named for the physical devices they represent and are usually created by using the ift(8) command. If no *aft_files* are listed, bb assumes standard device names. If *aft_files* are listed, they must be presented in the order in which the logical device is constructed. It is strongly recommended that you use the default names.

In its alternate form, you can use bb to locate a relative bad block on the special file, *special*, given the physical device, cylinder, head, and sector by using command-line option.

The *special* file must be a character or block special disk type device. Use the output of bb as input to the mkfs(8) and mkdmp(8) commands.

The bb command accepts the following options:

-g Lists good blocks, which are areas between the bad blocks.

-p Physical mode. Prints physical block numbers (sectors).

In its alternate form, bb requires the following information:

-D *device* Target device.

-C *cylinder* Target cylinder.

-H *head* Target head.

-S *sector* Target sector.

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 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 specify any file.

EXAMPLES

Example 1: An ASCII flaw table (aft) file is created using the ift command:

```
ift /dev/ift/0134 >/etc/aft/0134
```

Example 2: A typical use of the bb command is to initialize the dump device:

```
bb /dev/dsk/dump | mkdmp -b /dev/dsk/dump
```

Example 3: In its alternate form, bb is used to locate the logical block number of a given physical disk address:

```
bb -D 0230.0 -C 100 -S1 -H1 /dev/dsk/root.a
```

FILES

/etc/aft/*

SEE ALSO

ift(8), mkdmp(8), mkfs(8)

aft(5), dsk(4), pdd(4) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

NAME

bds – Bulk data server

SYNOPSIS

bds [*options*] ...

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

bds is a user level server, implemented as an enhancement to NFS, that provides direct I/O capabilities over the network. bds can be thought of as a remote DMA server: read/write requests are passed from client to the server; the server performs the request on the client's behalf. bds has no inherent block size limitation (other than the server DMA limit), and does not pass data through the file system caches on either the server or the client.

bds is normally started out of the startup scripts at boot time.

The bds command accepts the following options:

-*debug* Turns on debugging (sent to stderr).

-*devnull* Discards data written to the BDS server. Used for network debugging and performance testing only.

-*devzero* Generates zero filled data for read requests. Used for network debugging and performance testing only.

-*log* Logs open and close requests to stderr. close requests print out performance statistics.

-*touch* Touches the data after each I/O operation. Simulates local file system overhead when used with debugging and performance testing options.

CAUTIONS

bds binds to a fixed port number (port 2050). If the server is killed while there is an active connection, the port iwll remain "busy" until TCP determines there are no stray packets coming. Kill client applications before restarting the server.

BUGS

bds does not update file attributes during read/write operations. Applications that depend on immediate attribute updates after reads/writes may display unexpected behavior.

SEE ALSO

mount(8)

NAME

bmap – Identifies a block on a given file system

SYNOPSIS

```
/etc/bmap bno fsdev
/etc/bmap -p pbno pdev
/etc/bmap -p [-d] cyl trk sec pdev
```

IMPLEMENTATION

Cray PVP systems except CRAY J90 series and CRAY EL series

DESCRIPTION

The bmap command discovers the current use of a given block on a given file system. This is useful when an uncorrectable disk error occurs on a block and it needs to be put in a flaw table.

If a system administrator knows to what file a block belongs, the administrator can inform the file owner of the loss or restore it from backups.

You can specify the following operands on the command line. If a number given for the *bno*, *pbno*, *cyl*, *trk*, or *sec* operand begins with 0, it is assumed to be an octal number.

bno	Logical block number relative to file system.
fsdev	Block special device name containing file system.
-p	Indicates physical block is being specified.
-d	Indicates that disk relative sector is being given.
pbno	Physical block (512 words) offset on device.
pdev	Physical device name.
cyl	Cylinder.
trk	Track
sec	Sector

The specified block can be any of the following types:

Type	Description
boot block	Unused
super block	File system super block
copy N of super block	Copy N of file system super block
dynamic super block	File system dynamic block
shared filesystem block	(For 9 systems) Shared file system block (SFS only)

`inode region map block`	File system inode bit map block
`an inode block for inodes P.R.N - M`	File system block that contains inodes in partition P, region R, numbers N through M
`filesystem bit map block`	File system block that contains bit map of available space
`free block`	Unassigned file system blocks
`inode`	Inode for the named file
`data block`	Data block for the named file
`indirect data block`	Block of address extents for the named file
`ACL block`	Access control list (ACL) block for the named file
`bad block`	Flawed block belonging to inode 0
`PAL block`	Privilege assignment list (PAL) block for the named 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 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.

EXAMPLES

In the following example, block 15,944 is the ACL block for the file `test.out`. The output from the command is as follows:

```
% bmap 15944 /dev/dsk/scr100

#
block 15944 is ACL block for file ./test.out
physical address:  49-A1-22 cylinder 01270 track 03 sector 032 block 234008
(After adding spiral offset:  49-A1-22 cylinder 01270 track 03 sector 032)
#
```

SEE ALSO

`dmap`(8), `fsmap`(8)

General UNICOS System Administration, Cray Research publication SG–2301

NAME

brc, bcheckrc, rc, rc.mid, rc.pre, rc.pst – Invokes system initialization shell scripts

SYNOPSIS

```
/etc/bcheckrc
/etc/brc
/etc/rc
/etc/rc.mid
/etc/rc.pre
/etc/rc.pst
```

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

These shell scripts are executed by init(8) using entries in /etc/inittab when changing the system out of single-user mode.

The brc and bcheckrc shell scripts execute only the first time the system is changed out of single-user state (that is, after a reboot). The rc script can be executed whenever init(8) changes the run-level of the system. Usually, init(8) executes rc when changing from single-user mode to run-level 2, the usual multiuser run-level.

As released by Cray Research, the bcheckrc, brc, and rc scripts are not intended to be modified directly. The actions of these scripts can be controlled by setting environment variables in the /etc/config/rcoptions file. Additionally, the rc script calls several user-customizable subsidiary scripts, to allow for local configuration actions that fall outside of the usual actions performed by rc. (If you find it necessary to modify the bcheckrc, brc, or rc scripts to accomplish your necessary system startup actions, notify Cray Research so that the modification can be considered for inclusion in future released versions of these scripts.)

The bcheckrc script performs necessary system checks before the rc script can be executed. The bcheckrc script will set the system date and time as controlled by the RC_DATE environment variable. The bcheckrc script will check file system consistency by calling the mfsck(8) utility. The RC_FSCK environment variable controls whether the mfsck -u option will be used for this check. The bcheckrc script should not be used to start processes, because those processes will be killed and not restarted during the subsequent system shutdown or startup.

The brc script is intended for use in initializing hardware devices; it also copies raw core dump files to a regular UNICOS file in a separate file system, by executing the coredd(8) shell script. The brc script should not be used to start processes, because those processes will be killed and not restarted during the subsequent system shutdown or startup.

The rc shell script performs the following functions:

- Reads the environment variables in the /etc/config/rcoptions file to determine which actions it should perform (see the ENVIRONMENT VARIABLES section).
- Saves the existing contents of the startup output log file (specified by the RC_LOG environment variable; default is /etc/rc.log) to the same log file name with a suffix of .PREF (for example, /etc/rc.log.PREV).
- Logs the time of system startup and the version of rc performing the startup.
- Starts the file system error log daemon fslogd(8).
- Executes the local script rc.pre to perform any local initialization configured by the system administrator. (At this point in system startup, no file systems, including the /usr file system, have been mounted. Thus, any initialization performed in the rc.pre script must **not** attempt to access any files, commands, or directories that are not on the root file system. This includes the commands in /usr/bin and /usr/ucb. For initialization that occurs after the file systems have been mounted, see rc.mid below.)
- When coming from single-user mode, unmounts all currently-mounted file systems (to clear the way for the later file system mounts).
- Mounts the /tmp file system, using the device specified by the TMPDEV environment variable. If no device is specified by the TMPDEV environment variable, the rc script will use the device specified by the first entry for /tmp in the /etc/fstab file. As controlled by the RC_MKTMP environment variable, the rc script may first reinitialize the /tmp file system by using the mkfs(8) command.
- Mounts the /usr file system, using the device specified by the USRDEV environment variable. If no device is specified by the USRDEV environment variable, the rc script will use the device specified by the first entry for /usr in the /etc/fstab file.
- Mounts the /usr/tmp file system, using the device specified by the USRTMPDEV environment variable. If no device is specified by the USRTMPDEV environment variable, the rc script will use the device specified by the first entry for /usr/tmp in the /etc/fstab file. As controlled by the RC_MKTMP environment variable, the rc script may first reinitialize the /usr/tmp file system by using the mkfs(8) command.
- Mounts the user file systems specified in the /etc/fstab file and as controlled by the CRI_RC for the entries in that file.
- Removes world access to the /usr/src directory.
- Mounts the /proc file system.
- Adjusts the SECDED processing parameters to reduce system overhead from memory errors.
- Initializes the logical device cache by executing the ldcache(8) command with input from the /etc/config/ldchlist file.
- Initializes security by starting the security log daemon, setting security wildcard files, and initializing the network access lists.

- Preserves any interrupted vi(1) or ex(1) sessions.
- Executes the local script rc.mid to perform any local initialization configured by the system administrator. (This is the first local initialization script that is called after the user file systems have been mounted.)
- Performs administrative cleanup by saving mail from any system dump that was collected, removing account and mail queue lock files, and managing /usr/adm/sulog, the cron(8) log file (location specified by the RC_CRONLOGDIR environment variable; default is /usr/lib/cron), and the NQS log file (specified by the RC_NQSLOGFILE environment variable; default is /usr/spool/nqs/log).
- Initializes the /etc/wtmp file.
- Starts system accounting as controlled by the RC_ACCT environment variable.
- Starts the system activity daemon sadc(8) as controlled by the RC_SADC environment variable.
- Starts the system daemons in the SYS1 by using the sdaemon(8) utility.
- Initializes system networking by calling the netstart(8) script, as controlled by the RC_NET environment variable.
- Starts the system daemons in the SYS2 by using the sdaemon(8) utility.
- Executes the local script rc.pst to perform any local initialization configured by the system administrator.
- Logs the time of multiuser startup in the /etc/boot.log file.
- Logs successful completion of system startup.

The rc script can be used for several run-level states. Use the who(1) command to get the run-level information.

The rc.pre, rc.mid, and rc.pst scripts are called by the rc script at various points to allow for local startup initialization that can not be accomplished through the existing mechanisms in the rc script. The rc script passes the name of the startup output log file to each of the rc.pre, rc.mid, and rc.pst scripts, so that these scripts can then explicitly append any necessary output to the log file at the discretion of the person responsible for maintaining those scripts.

ENVIRONMENT VARIABLES

Most of the actions performed by the various system startup scripts are controlled by environment variables that are read in from the /etc/config/rcoptions file. Typically, the following values for a controlling environment variable specify whether or not the script will perform the action:

Value	Description
YES	Perform the action
NO	Do not perform the action
ASK	Prompt the operator for whether to perform the action

Additionally, an action can be tailored to specific run-levels by specifying a *decision string* with the following format:

runlevels=*action*[:*runlevels*=*action* ...]

runlevels is a concatenation of the run-level characters for which the specified *action* will be used. The *runlevels*= portion can be omitted to specify all run-levels. For example, the following decision string specifies that the action will be performed for run-levels 2 and 3, the operator should be prompted for run-levels 4 and 5, and the action should **not** be performed for any other run-levels:

```
23=YES:45=ASK:NO
```

Various other environment variables are used not to control an action, but to specify a device name, log file name, and so on.

The startup scripts use the following environment variables:

Script	Description
DUMPFS	Specifies the device name of the file system to be used for copying system core dumps. The default value is core.
DUMPMPT	Specifies the mount point to be used when copying system core dumps.
DUMPDIR	Specifies the directory in which the dump directory will be created when copying system core dumps.
RC_ACCT	Controls whether the rc script will start system accounting.
RC_CONTERR	Controls whether the rc script will continue system startup despite errors.
RC_CRONLOGDIR	Specifies the directory where the cron(8) log file is stored. The default value is /usr/lib/cron.
RC_DATE	Controls whether the bcheckrc script sets the system date and time.
RC_FSCK	Controls whether the bcheckrc script uses the mfsck -u option for file system checks.
RC_LDCH	Controls whether the rc script will activate the logical device cache using the ldcache(8) command.
RC_LOG	Specifies the file in which the rc script will save startup output. The default value is /etc/rc.log. If the value is null, all startup output will be sent to the system console (/dev/console).
RC_MKTMP	Controls whether the rc script will initialize the /tmp file system (using the mkfs(8) command).
RC_MKUTMP	Controls whether the rc script will initialize the /usr/tmp file system (using the mkfs(8) command).

RC_NET	Controls whether the rc script will call the netstart(8) script to start the networks.
RC_NQSLOG	Controls whether the rc script will clear the NQS log file.
RC_NQSLOGFILE	Specifies the name of the NQS log file. The default value is /usr/spool/nqs/log.
RC_SADC	Controls whether the rc script will start the system activity daemon sadc(8).
RC_SECHIGH	Specifies the upper security level used when initializing the /tmp or /usr/tmp file systems. The default value is 0. On systems that use nonzero security labels, this value should be set to 16, or if syshigh/syslow security labels are used, this value should be set to syshigh.
RC_SECLOW	Specifies the lower security level used when initializing the /tmp or /usr/tmp file systems. The default value is 0. On systems that use syshigh/syslow security labels, this value should be set to syslow.
RC_SECMASK	Specifies the security compartment mask used when initializing the /tmp or /usr/tmp file systems. The default value is 0. On systems that use nonzero security labels, this value should be set to 07777777777777777777 (low-order 63 bits).
RC_USRMNT	Controls whether the rc script will mount the user file systems.
TMPDEV	Specifies the device name of the file system that the rc script will mount on /tmp.
TMPOPTS	Specifies additional options for the mkfs(8) command that will be used to initialize the /tmp file system. This can be used to specify file system quotas for /tmp.
USRDEV	Specifies the device name of the file system that the rc script will mount on /usr.
USRTMPDEV	Specifies the device name of the file system that the rc script will mount on /usr/tmp.
USRTMPOPTS	Specifies additional options for the mkfs(8) command that will be used to initialize the /usr/tmp file system. This may be used to specify file system quotas for /usr/tmp.

FILES

/etc/boot.log	Logs the times of successful multiuser startups.
/etc/config/ldchlist	Initialization file for logical device cache (ldcache(8)).
/etc/config/rcoptions	Contains environment variables for configuring startup actions.

`/etc/fstab`	File system mount entries. Used by the `rc` script to determine the `/tmp` and `/usr/tmp` file systems, if not specified by environment variables.
`/etc/rc.log`	Default log file for system startup output and information.

SEE ALSO

`coredd`(8), `cron`(8), `fslogd`(8), `init`(8), `ldcache`(8), `mfsck`(8), `mkfs`(8), `netstart`(8), `sadc`(8), `sdaemon`(8), `shutdown`(8)

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

`fstab`(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

captoinfo – Converts a termcap description into a terminfo(5) description

SYNOPSIS

/usr/bin/captoinfo [-v...] [-1] [-w *width*] *file* ...

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The captoinfo command looks in *file* for termcap descriptions. For each description that is found, an equivalent terminfo(5) description is written to standard output, along with any comments found. A description that is expressed as relative to another description (as specified in the termcap tc= field) will be reduced to the minimum superset before being output.

The captoinfo command accepts the following options:

-v... Prints tracing information on standard error as the program runs. Specifying an additional -v option will cause more detailed information to be printed.

-1 Causes the fields to print, one to a line. Otherwise, the fields will be printed several to a line with a maximum width of 60 characters.

-w *width* Changes the output to *width* characters.

file ... If no *file* is specified, the TERMCAP environment variable is used for the file name or entry. If TERMCAP is a full path name to a file, only the terminal whose name is specified in the TERM environment variable is extracted from that file. If the TERMCAP environment variable is not set, the /etc/termcap file is read.

NOTES

Certain termcap defaults are assumed to be true. For example, the bell character (terminfo bel) is assumed to be ^G. The line-feed capability (termcap nl) is assumed to be the same for both cursor_down and scroll_forward (terminfo cud1 and ind, respectively). Padding information is assumed to belong at the end of the string.

The algorithm used to expand parameterized information for termcap fields such as cursor_position (termcap cm, terminfo cup) will sometimes produce a string which, though technically correct, may not be optimal. In particular, the rarely used termcap operation %n will produce strings that are especially long. Most occurrences of these nonoptimal strings will be flagged with a warning message and may need to be recoded by hand.

The short two-letter name at the beginning of the list of names in a `termcap` entry, a hold-over from an earlier version of the UNIX system, has been removed.

WARNINGS

The `captoinfo` command should be used to convert `termcap` entries to `terminfo`(5) entries because the `termcap` database (from earlier versions of UNIX System V) may not be supplied in future releases.

FILES

`/usr/lib/terminfo/?/*`	Compiled terminal description database

SEE ALSO

`infocmp`(8), `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

chargefee – Charges a fee to a user

SYNOPSIS

/usr/lib/acct/chargefee *login-name account-name fee*

/usr/lib/acct/chargefee -d *login-name account-name fee*

/usr/lib/acct/chargefee -D -d *login-name fee*

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The chargefee command charges a fee to a user. chargefee writes a record to /usr/adm/acct/day/fee; the runacct(8) main accounting shell procedure or the csaperiod(8) command merges this record with other accounting records.

The chargefee command accepts the following options:

-d	Outputs the fee records in cacct format. When using Cray Research system accounting (CSA), you should specify this option.
-D	Uses the user's default account ID.
login-name	The user name from the /etc/udb file.
account-name	The account ID field from the /etc/udb file.

SEE ALSO

csaperiod(8), runacct(8)

NAME

chmem – Changes the system's notion of physical memory size

SYNOPSIS

Current:

```
/etc/chmem -s
/etc/chmem -m reserve|rsv [-s]
/etc/chmem -m release|rls [-s]
```

Obsolescent (may not be supported in future releases):

```
/etc/chmem [[+]inc[m]]
/etc/chmem [[+]inc[M]]
/etc/chmem [[-]inc[m]]
/etc/chmem [[-]inc[M]]
```

IMPLEMENTATION

Cray PVP systems

DESCRIPTION

As of the UNICOS 8.3 release, the functionality of the chmem command changed. The older functionality, however, is still supported. This man page lists the current command usage under the heading "Current," and the older usage under the heading "Obsolescent."

Current

The chmem command displays the sizes of physical memory areas and dynamically configures maintenance memory at run time. Dynamic maintenance memory management is supported on CRAY T90 systems only. The chmem command accepts the following options:

`-s` Displays the sizes of the currently configured memory areas that comprise physical memory. If a memory area (as described above) is not currently configured, it is not displayed.

If the `-m` option is specified, the displayed sizes reflect the memory status after the operation designated by the `-m` option has been performed.

`-m reserve|rsv`
Reserves maintenance memory by reducing the overall size of configured physical memory by the amount required for diagnostics. The required space comes from user memory.

`-m release|rls`
Releases maintenance memory by restoring the overall size of configured physical memory. The space used for maintenance memory is returned back to user memory.

Physical memory can contain the following areas (the first two always exist; the last three are configuration-dependent and may not exist):

Memory Areas	Description
kernel memory	Three separate areas that are not necessarily contiguous: Space allocated at build time. Space allocated at boot time. Space allocated at run time.
user memory	Area where user processes reside.
ramdisk	Memory resident disk.
guest memory	Area where guest operating systems reside (mutually exclusive from downed memory).
downed memory	Area that was formerly used by diagnostics. This area is an artifact of the archaic form of `chmem` (mutually exclusive from guest memory).
maintenance memory	Area used by diagnostics.

Obsolescent

The `chmem` command is the administrator command interface to the `chmem`(2) system call. When invoked without the size specification argument (*inc*), `chmem` displays the value of the system's current notion of physical memory.

inc The *inc* operand can be preceded immediately by a `+` or `-` indicating a relative adjustment; otherwise, *inc* is taken as absolute size. The *inc* operand can be immediately followed by an `m` or `M`, indicating that *inc* is being expressed in Mwords.

Only an appropriately authorized user can adjust the system's current notion of physical memory.

NOTES

Current

To use the `chmem` command, a user must have read permission for `/dev/mem`.

Obsolescent

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.

EXAMPLES

Obsolescent

The following command line reduces the system's notion of physical memory by 8 Mwords:

```
chmem -8m
```

The following command line increases the system's notion of physical memory by 4 Mwords:

```
chmem +4m
```

SEE ALSO

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

NAME

chroot – Changes the root directory and executes a command

SYNOPSIS

/bin/chroot *newroot command*

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The chroot command executes *command* relative to *newroot*. The meaning of any initial slashes (/) in path names is changed to *newroot* for a command and any of its children. Furthermore, the initial working directory is *newroot*.

The following command line creates the x file relative to the original root, rather than the new one:

```
chroot newroot command >x
```

Only an appropriately authorized user can use this command.

The new root path name is always relative to the current root: even if a chroot is currently in effect, the *newroot* operand is relative to the current root of the running process.

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. File access and shell redirected I/O is subject to security label restrictions.

If the PRIV_SU configuration option is enabled, the super user or a user with the CHROOT permbit is allowed to use this command.

CAUTIONS

Use extreme caution when referencing special files in the new root file system, because files above the newly defined root in the file structure will become inaccessible.

SEE ALSO

udbgen(8)

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

NAME

ckdacct – Checks the size of daemon accounting files

SYNOPSIS

/usr/lib/acct/ckdacct [-n *blocks*] *daemons*

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The ckdacct command checks the size of the daemon accounting files and checks the amount of free space on the file system containing the /usr/adm/acct directory. If the size of the daemon accounting file exceeds 500 blocks (default) or exceeds the specified number of blocks, ckdacct starts a new accounting file by invoking the turndacct(8) command with the switch operand.

ckdacct also ensures that the ACCT_FS file system contains at least MIN_BLKS free blocks. If there is not this much free space, ckdacct turns off accounting for the specified daemons by invoking turndacct(8) with the off operand. ckdacct calls turndacct(8) with the on operand to re-enable daemon accounting when at least MIN_BLKS free blocks are available.

ACCT_FS is a parameter which defines the file system on which /usr/adm/acct resides and is defined in the accounting configuration file /etc/config/acct_config. The MIN_BLKS parameter also is defined in this file.

This feature is sensitive to the frequency at which ckdacct is executed. You should run ckpacct(8) periodically by using the cron(8) command.

The ckdacct command accepts the following option, argument, and operand:

-n *blocks* Specifies the maximum size (in blocks) to which the daemon accounting files can grow before they are switched. The default is 500 blocks.

daemons Specifies a list of daemons for which the accounting file sizes are checked. Daemon names are separated by white space. Valid daemon names are nqs, tape and socket.

In the released template of the accounting configuration file, /etc/config/acct_config, ACCT_FS is set to /usr. If this is not correct for your system, you must define ACCT_FS properly in /etc/config/acct_config.

EXAMPLES

The following example is a suggested entry for the /usr/spool/cron/crontabs/root file so that cron(8) automatically runs ckdacct on the hour:

```
0   *   *   *   *   /usr/lib/acct/ckdacct nqs tape
```

FILES

`/etc/config/acct_config`	Accounting configuration file
`/usr/adm/acct/day`	Directory that contains current daemon accounting files
`/usr/spool/cron/crontabs/root`	Root `crontab`(1) file

SEE ALSO

`acctsh`(8), `cron`(8), `csa`(8), `turndacct`(8)

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

UNICOS Resource Administration, Cray Research publication SG–2302

NAME

ckpacct – Checks the size of the process accounting file

SYNOPSIS

/usr/lib/acct/ckpacct [*blocks*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The ckpacct script checks the size of the process accounting file /usr/adm/acct/day/pacct and checks the amount of free space on the file system containing the /usr/adm/acct directory. If the size of the pacct file exceeds 500 blocks (default) or exceeds the specified number of blocks, ckpacct starts a new accounting file by invoking the turnacct(8) command with the switch operand.

ckpacct also ensures that the ACCT_FS file system contains at least MIN_BLKS free blocks. If there is not this much free space, ckpacct turns off process accounting by invoking turnacct(8) with the off operand. ckpacct calls turnacct(8) with the on operand to re-enable process accounting when at least MIN_BLKS free blocks are available.

ACCT_FS is a parameter which defines the file system on which /usr/adm/acct resides and is defined in the accounting configuration file /etc/config/acct_config. The MIN_BLKS parameter also is defined there.

This feature is sensitive to the frequency at which ckpacct is executed. You should run ckpacct periodically using the cron(8).

The ckpacct script accepts the following operand:

blocks Specifies the maximum size (in blocks) to which the process accounting file can grow before it is switched. The default is 500 blocks.

In the released template of the accounting configuration file, /etc/config/acct_config, ACCT_FS is set to /usr. If this is not correct for your system, you must define ACCT_FS properly in /etc/config/acct_config.

EXAMPLES

The following example is a suggested entry for the /usr/spool/cron/crontabs/root file so that cron(8) automatically runs ckpacct on the hour:

```
0   *   *   *   *   /usr/lib/acct/ckpacct
```

FILES

`/etc/config/acct_config`	Accounting configuration file
`/usr/adm/acct/day/pacct*`	Process accounting files
`/usr/spool/cron/crontabs/root`	Root `crontab`(1) file

SEE ALSO

`acctsh`(8), `cron`(8), `csa`(8), `turnacct`(8)

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

UNICOS Resource Administration, Cray Research publication SG–2302

NAME

`cleantmp` – Deletes job temporary directories

SYNOPSIS

```
/etc/cleantmp username path
/etc/cleantmp  all
```

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

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

username path When specified with a *username* and *path*, deletes job temporary directories and temporary directories created by `tmpdir`(1). `cleantmp` is called by `init`(8) and the Network Queuing System (NQS).

`all` When specified from single-user mode, deletes job temporary directories and temporary directories created by `tmpdir`(1) that were not deleted normally due to a system crash or another problem.

NOTES

The `cleantmp` command should not be used in multiuser mode.

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.

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

MESSAGES

The `cleantmp` command prints its error messages on the system console.

FILES

`${TMPDIR}/.tmpdir` File that contains list of temporary directories.

`${TMPDIR}/.tmpdir[a-z]` On a UNICOS system using multilevel security labels, the files created by `tmpdir`(1) have names ending in a letter `a` through `z`. This naming convention enables a user to successfully change security levels and/or compartments (up to a maximum of 26 times) and still access the temporary directory feature.

SEE ALSO

`init`(8)

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

NAME

`cll` – Lists or resets the login failure attempts field in user database (UDB)

SYNOPSIS

```
/etc/cll -r user
/etc/cll -R
/etc/cll -l user
/etc/cll -L
```

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `cll` command is used by an appropriately authorized user to list or reset the failed login attempts for one user or all users. The number of failed login attempts is recorded in the `logfails` field of the user's entry in the user database (UDB).

The `cll` command accepts the following options (at least one of these options must be specified on a command line):

`-r` *user* Resets the login failed attempts field for the specified *user*. This option can be used alone or with the `-R` option.

`-R` Resets the login failed attempts field for all users. This option can be used alone or with the `-r` option.

`-l` *user* Lists the login failed attempts field for the specified *user*. This option can be used alone or with the `-L` option.

`-L` Lists the login failed attempts field for all users. This option can be used alone or with the `-l` option.

To lock out a user, the following conditions must be met:

- The `MAXLOGS` parameter must be greater than 0.
- The `DISABLE_ACCT` parameter must be enabled.
- The `DISABLE_TIME` parameter must be greater than 0.

The `MAXLOGS`, `DISABLE_ACCT`, and `DISABLE_TIME` parameters are defined in the `uts/cf.SN/config.h` file.

If these conditions are met and `logfails` equals or exceeds the value defined by `MAXLOGS`, the user is locked out of the system until the `logfails` field is reset by an appropriately authorized user or the number of seconds specified by the `DISABLE_TIME` parameter has elapsed.

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 action shown:

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

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

FILES

`/etc/udb`	User database file
`/etc/udb.public`	Public user database file
`/etc/udb_2/udb.index`	Public extension index file
`/etc/udb_2/udb.priva`	Private field extension file
`/etc/udb_2/udb.pubva`	Public field extension file
`uts/cf.SN/config.h`	Kernel parameter definition file

SEE ALSO

`udbgen`(8)

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

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

General UNICOS System Administration, Cray Research publication SG–2301

NAME

cm – Moves a spindle in IOS model E disk configuration

SYNOPSIS

cm [-D *ddir*] [-P *pdir*] [-a *value*] [-d] [-r] [-V] [-v] *from to*

IMPLEMENTATION

Cray PVP systems with I/O subsystem model E

DESCRIPTION

The cm command (configuration move) moves the configuration of slices or partitions from physical address *from* to physical address *to*. Additional options allow you to specify an ASCII summary of the physical address, move the data, and produce debug listings.

The program recognizes two directories: /dev/dsk, the location of mountable names, and /dev/pdd, the location of the physical disk devices. The scan of the disk farm begins in the /dev/dsk directory. The scan progresses recursively down the logical device definitions to the physical device definitions, usually in /dev/pdd, that contain the channel and unit numbers. This algorithm ignores physical devices that are not a part of a mountable logical device.

The cm command accepts the following operands, which are required:

from, *to* Physical addresses. In most cases, these are numeric character strings that are converted to internal values by using standard conventions for numeric conversion (0 x 40 = 0100 = 64). For DD-60 devices, a unit number may follow a period -- 0232.3.

To save the spindle configuration summary in an ASCII readable file, the *from* or *to* values can be file names.

from and *to* values of - correspond to stdin or stdout.

The cm command accepts the following options:

-D *ddir* Replaces the /dev/dsk as the name of the directory where the device scan begins recursively processing devices. This option is intended for debug purposes.

-P *pdir* Replaces /dev/pdd as the directory name where the physical devices are placed when a mknod is performed. Note that the physical devices are found by following logical device specifications from /dev/dsk. This option is intended for debug purposes.

-a *value* The -a option accepts two values: i and o. If the value is i, the *from* argument names an ASCII input file that contains the configuration to be installed at the *to* address. If the value is o (or not-i), the *to* argument names a file that will become the ASCII output description of the file.

`-d` Indicates that data should be moved. If the two arguments to the `cm` program are physical disk addresses, the movement of data will be transparent to the user when the file systems are remounted. If a version of the `-a` option is in effect, the data associated with `/dev/pdd/name_1` will be copied to/from `name_1` in the current directory. The data copy routine was designed to deal with failing input partitions; it will seek over areas where the read fails with EIO.

`-r` Removes the configuration at the *from* address. If the `-a` option, in either variety has not been specified, the removal of the *from* configuration is unavoidable. This flag is meaningful only if `-a o` is specified.

`-V` Indicates verbose no-operation.

`-v` Indicates verbose operation. Each `mknod`, `rename`, `unlink`, and `dd` is announced before execution.

NOTES

Use the following command to survey the situation:

```
cm -ao 0nnn.k -
```

Work carefully. This utility has the potential to deconfigure everything in the disk farm.

EXAMPLES

Example 1: Removes all devices at address 0204.5:

```
cm -r -a o 0204.5 /dev/null
```

Example 2: Replaces address 0206 with 0210:

```
cm -d 0206 0210
```

Example 3: Uses scratch files in the current directory to hold information while the device at 0206.6 is being replaced:

```
cm -a o -d 0206.6 savit
# pause for disk replacement
# pause for new flaws on 0206.6
cm -a i -d savit 0206.6
```

Example 4: Changes spindle 0234.7 with 0236.6, using scratch files in the current directory to hold the data:

```
cm -rdao 0234.7 xxfile
cm -rd   0236.6 0234.7
cm -dai  xxfile 0236.6
```

NAME

coredd – Automatically copies raw core dump files to a regular UNICOS file in a separate file system

SYNOPSIS

/etc/coredd

IMPLEMENTATION

Cray PVP systems

DESCRIPTION

The coredd shell script is executed by brc(8) when changing the system out of single-user mode (see inittab(5)). Dump file system (Dumpfs), dump mount point (Dumpmpt), and dump directory (Dumpdir) are passed to coredd from brc. They are set by using the configure system -> special system device definitions -> menu in the install tool. The default block device is /dev/dsk/core; the default file system and dump directory is /mnt. The Dumpfs dump file system is mounted on the Dumpmpt dump mount point. coredd calls the cpdmp(8) command, which is executed to determine whether a dump exists. If a dump exists, coredd performs the following operations:

- Creates a subdirectory in the /Dumpmpt/Dumpdir/ directory to contain the system dump information. The name of this subdirectory is a time stamp of when the dump was moved, in the format /Dumpmpt/Dumpdir/*mmddhhmm* (for example, /<mnt>/10120823).
- Calls cpdmp to copy the dump to the /Dumpmpt/Dumpdir/*mmddhhmm* subdirectory in a file called dump.
- Copies the version of the UNICOS operating system (that is, the kernel) that was running at the time of the crash. This is put in the /Dumpmpt/Dumpdir/*mmddhhmm* subdirectory in a file called unicos.
- Copies the version of crash(8) to be used with the dump; this is put in the /Dumpmpt/Dumpdir/*mmddhhmm* subdirectory in a file called crash.

If an error is detected in processing the dump, a message is sent to the console asking for operator intervention to move the dump, unicos, and crash files after the system is up in multiuser mode.

MESSAGES

The coredd script has been changed to mount /dev/dsk/core onto /mnt instead of /core. At boot time, this causes a warning message, as follows:

```
<core> mounted as <mnt>
```

This message should be ignored.

SEE ALSO

`brc`(8), `cpdmp`(8), `crash`(8)

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

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

NAME

cpdmp – Processes one or more IOS SYSDUMP areas

SYNOPSIS

/etc/cpdmp [-n] [-c] [-l *logfile*] [-i *rawdump*] [-o *dumpfile*]

IMPLEMENTATION

Cray PVP systems with IOS model E

DESCRIPTION

The cpdmp command processes one or more I/O subsystem (IOS) SYSDUMP areas, combining them into one system dump file. It is executed by the coredd(8) script when changing the system out of single-user mode (see inittab(5)).

The system dump contains a header that details the memories and ranges that were dumped. cpdmp reformats the dump and header to a format recognized by crash(8).

The cpdmp command accepts the following options:

-n	Does not copy system dump; sets exit code to reflect SYSDUMP status.
-c	Clears the dump copied flag; dump is not moved.
-l *logfile*	Specifies the log file name. The default is /etc/dump.log.
-i *rawdump*	Specifies the raw dump device. The default is /dev/pdd/dump.
-o *dumpfile*	Specifies the system dump file name. The default is /core.sys.

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 specify any file.

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

EXIT STATUS

The cpdmp command generates the following exit codes:

Exit Code	Description
2	No SYSDUMP area to process.
1	Fatal error encountered; unable to process input or generate system dump file.
0	SYSDUMP successfully processed or available for processing (-n option).

FILES

`/dev/pdd/dump`	Device from which to copy the dump image
`/core.sys`	File to which dump image is written
`/etc/dump.log`	File containing time, memories, and ranges for each dump

SEE ALSO

`coredd`(8), `crash`(8)

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

`mfsysdmp`(7) (online only)

NAME

cpset – Installs object files in binary directories

SYNOPSIS

/usr/bin/cpset [-o] [-n] *object destination* [*mode* [*owner* [*group*]]]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The cpset command installs the specified *object* file at the given *destination*. If *destination* already exists, it is removed before *object* is installed. If *destination* is a directory, *object* is installed as a file named *destination*/*object*. The *mode*, *owner*, and *group* of the destination file can be specified on the command line. If this data is omitted, two results are possible:

1. If you are an appropriately authorized user, the following defaults are provided:

   mode = 0755
   owner = bin
   group = bin

2. Otherwise, the default owner and group of the destination file are yours.

The cpset command accepts the following options:

-o Forces cpset to move *object* to OLD*object* in the *destination* directory before installing the new *object*.

-n Tells cpset to handle files that are "text busy" (ETXTBSY) by linking the files to OLD*object* in the destination directory before installing the new object. If the OLD*object* is also busy, cpset iteratively increments the name to OL2*object*, continuing to increment until a file name that is not busy is encountered.

The cpset command uses the /usr/src/destinations file to determine the final destination of a file. The *destination* file contains pairs of path names separated by spaces or tabs. The first name is the official destination (for example, /bin/echo). The second name is the new destination. For example, if echo is moved from /bin to /usr/bin, the entry in /usr/src/destinations would be as follows:

```
/bin/echo      /usr/bin/echo
```

When the actual installation occurs, cpset verifies that the old path name does not exist. If a file exists at that location, cpset issues a warning and continues. This file does not exist on a distribution tape; it is used by sites to track local command movement. The procedures used to build the source must define the official locations of the source.

If you are using `cpset` to move system binaries while generating a system (for example, while programming in a shell script or makefile that you have written), you should move to `$ROOT/`*directory/file* rather than to */directory/file*.

NOTES

`cpset` does not set access control lists (ACLs). ACLs must be set independently by using the `spset`(1) 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
`id`	`cpset` attempts to set file owner, group, and mode to `bin:bin:0755`.

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`	Allowed to set file owner, group, and file mode to `bin:bin:0755`.

If the `PRIV_SU` configuration option is enabled, the super user or the user `bin` who belongs to the `bin` group is allowed to set the file owner, group, and mode to `bin:bin:0755`. If the user's effective ID is less than 100, `cpset` attempts to set the file owner, group, and mode to `bin:bin:0755.`

EXAMPLES

The following examples have the same effect (assuming that the user is appropriately authorized). The `echo` file is copied into `/bin` and is given access permissions of `0755`, owner ID of `bin`, and group ID of `bin`.

```
cpset echo /bin 0755 bin bin
cpset echo /bin
cpset echo /bin/echo
```

SEE ALSO

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

NAME

cpu – Selects, dedicates, and changes mode bits per CPUs

SYNOPSIS

```
/etc/cpu mask | -n newmask [-D] [-r] [-m mode] command [arguments]
/etc/cpu [mask | -n newmask] [-D] [-r] [-m mode] -p pids
/etc/cpu mask | -n newmask -d | -u
/etc/cpu -d
/etc/cpu -t ticks
/etc/cpu mask | -n newmask -s | -S
/etc/cpu mask | -n newmask [-r] [-m mode] [-p pids] [command [arguments]]
```

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The cpu command allows you to select one or more processors in which to run some designated processes.

The processors are chosen according to *mask* (obsolescent version) or -n *newmask* (current version).

The new style string consists of a comma-separated list (-n 0,3,4, ... *number-of-CPUs*) and/or a range designation (-n 0-5,10-14, ...).

Support for the old style is preserved for compatibility. The old style should be a string that contains letters (a through z) or numbers (0 through *7*). Of course, entries are limited to the number of CPUs on your system.

When the *command* operand is specified, cpu runs *command* (according to the specified arguments) on the chosen processors. When the -p option is specified, cpu runs each of the processes specified by *pid* on the chosen processors.

On Cray PVP systems, the *mask* argument is silently limited to the available processors. If all processors chosen are unavailable, the process is allowed to run on any processor.

The cpu command accepts the following option and operand on all Cray Research systems:

-n *newmask*
: Selects processors. The *newmask* string consists of a comma-separated list (-n 0,3,4, ... *number-of-CPUs*) and/or a range designation (-n 0-5,10-14 ...).

-p *pids*
: Specifies one or more process IDs with a comma-separated list or a spaces-separated list in quotes.

The cpu command accepts the following options and operands on CRAY Y-MP systems:

`-D` Dedicates the specified CPUs to the command or processes. The CPUs are freed when the process completes.

If used without a mask, any CPUs previously dedicated to the processes will no longer be dedicated to those processes.

`-d` If used with a mask, disables the CPUs specified by mask. The system does not let the last available CPU be disabled.

If used without a mask, displays the down CPUs. This option works on all Cray Research systems.

`-t` *ticks* Changes the system tick rate to the specified number of *ticks* per second. Valid values are 1 to 1000.

`-s` Disables scalar cache for CPUs specified by *mask* or *newmask*.

`-S` Enables scalar cache for CPUs specified by *mask* or *newmask*.

`-u` Enables the CPUs specified by mask.

`-r` Reports specific mode bits enabled or disabled.

`-m` *mode* Changes specific mode bits. Acceptable modes are as follows:

`monon` Monitor mode on (super user only).

`monoff` Monitor mode off (super user only).

`bdmon` Bidirectional memory on.

`bdmoff` Bidirectional memory off.

`emaon` Enables the EMA mode bit in the exchange package, which enables 24/32 bit address mode and the corresponding instructions.

`emaoff` Disables the EMA mode bit in the exchange package.

`avlon` Second vector logical on.

`avloff` Second vector logical off.

`fpeon` Floating-point interrupts on.

`fpeoff` Floating-point interrupts off.

`oreon` Operand range errors on.

`oreoff` Operand range errors off.

`icmon` Interrupt on correctable memory errors on.

`icmoff` Interrupt on correctable memory errors off.

`iumon` Interrupt on uncorrectable memory errors on.

`iumoff` Interrupt on uncorrectable memory errors off.

`immon` Interrupt on monitor mode on.

`immoff` Interrupt on monitor mode off.

`rpeon` Interrupt on register parity error mode on.

`rpeoff` Interrupt on register parity error mode off.

`scon` (CRAY T90 and CRAY J90 series) Scalar cache enabled/only runs on CPUs with cache

`scoff` (CRAY T90 and CRAY J90 series) Scalar cache disabled

The following modes are valid on CRAY T90 systems with IEEE floating point CPUs only:

`xion` IEEE floating point exceptional input interrupt on.

`xioff` IEEE floating point exceptional input interrupt off.

`nxon` IEEE floating point inexact interrupt on.

`nxoff` IEEE floating point inexact interrupt off.

`unfon` IEEE floating point underflow interrupt on.

`unfoff` IEEE floating point underflow interrupt off.

`ovfon` IEEE floating point overflow interrupt on.

`ovfoff` IEEE floating point overflow interrupt off.

`dvion` IEEE floating point divide by zero interrupt on.

`dvioff` IEEE floating point divide by zero interrupt off.

`nvion` IEEE floating point invalid interrupt on.

`nvioff` IEEE floating point invalid interrupt off.

`rm0on` IEEE rounding mode 0 on.

`rm0off` IEEE rounding mode 0 off.

`rm1on` IEEE rounding mode 1 on.

`rm1off` IEEE rounding mode 1 off.

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 I/O and access to specified processes are subject to security label restrictions.

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

On IEEE CRAY T90 systems, the fpeon and fpeoff modes may be used to set and clear the IEEE floating point mode flags. The fpeon mode will turn on a default set of the IEEE interrupt modes (ovfon, dvion and nvion). The fpeoff mode will turn off all IEEE interrupt modes. Specifying any of the IEEE modes along with fpeon or fpeoff is an error.

The IEEE rounding mode specifications are as follows:

rm0off rm1off	Round to nearest
rm0on rm1off	Round up
rm0off rm1on	Round to zero
rm0on rm1on	Round down

MESSAGES

cpu sends messages about a *pid* that is not valid and about a mask character that is not valid.

EXAMPLES

Example 1: The following example downs CPUs 0, 2, and 3:

```
cpu -n 0,2,3 -d
```

Example 2: This command undedicates all CPUs from process 53536:

```
cpu -D -p 53536
```

SEE ALSO

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

cpu(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

`crash` – Examines system core images

SYNOPSIS

`/etc/crash` [`-g` *name*] [`-s`] [*core_filename* [*namelist_filename*]]

IMPLEMENTATION

Cray PVP systems

DESCRIPTION

The `crash` command is an interactive utility for examining an operating system core image. It has facilities for interpreting and formatting the various control structures in the system and certain miscellaneous functions that are useful when perusing a dump.

The `crash` utility accepts the following options and operands

`-g` *name* Specifies the guest system name to operate on. A memory bias (guest load address) will be automatically added to all `crash` memory requests This option should be used only when a guest is active.

`-s` (Internal use only) Specifies whether the program is being executed from the command line or whether it has been called by `crash` on the host system. This parameter is used internally by `crash` and should not be used on the command line.

core_filename Specifies where the system image can be found. The default value of *core_filename* is `/dev/mem`, which lets you use `crash` without an operand to examine an active system. If you specify the system image file, it is assumed to be a system core dump and the default process is set to be that of the process active in the kernel at the time of the crash. This is determined by a value stored in a fixed location by the dump mechanism.

namelist_filename Specifies the binary matching *core_filename*. Its default value is `/unicos`.

Input to `crash` is typically of the following form:

command [*options*] [*slot numbers to be printed*] [`>` *file*] [`|` *command*]

If you specify no specific structure elements, all valid entries will be used. The following example would print the entire process table in standard format:

```
proc
```

When allowed, *options* modify the format of the printout. For example, the following command line prints process table slots 12, 15, and 3 in a long format:

```
proc - 12 15 3
```

Format data structure entries assume a decimal slot number (zero-based). Those commands that perform I/O with addresses assume an octal slot number.

The following subcommands are available for systems configured with Cray Research Distributed Computing Environment (DCE) Distributed File Service (DFS): `aggr`, `ccall`, `cct`, `ch`, `chtable`, `cm_conn`, `cm_serv`, `dcache`, `dfsmisc`, `dfsstat`, `fid`, `fshost`, `scache`, `sct`, `tkc`, `tkm`, `tkset`, `tpq`, and `volume`.

The `crash` command accepts the following commands:

`?` Prints synopsis of commands

`!` Escapes to shell

`q` Exits from `crash`

`.` Repeats the last command

`address` *table_name slot_number*
Aliases: `addr`
Specifies the address of the given table entry number. The address can then be used with the `od` subcommand.

`aggr` [`-`|`--`|`---`|`----`] [`-g`] [`-l`] [*addr*]
Alias: `ag`
(DCE DFS only) Displays aggregate entries (`struct aggr`) in the `ag_root` list.

`-`, `--`, `---`, `----`
Specifies verbose level; displays additional information in the output. Increasing the number of dashes (the `-` character) increases the amount of information that is displayed.

`-g` Produces a command and address in `grep`(1) format that can be used to display a specific structure.

`-l` Displays information about the lock structures contained in the structure(s) being displayed.

[*addr*] Limits the display to the data structure at the specified address. By default, all known data structures are displayed.

`aio` [*slot_numbers*]
Displays the specified `aio` (asynchronous I/O) entries. By default, all entries are displayed.

`bfreelist`
Displays the buffer cache free list.

`bfreemt`
Displays the buffer cache empty list.

`buf` [*buffer_headers*]
Aliases: `hdr`, `bufhdr`
Formats the specified system buffer headers.

`buffer` [*format*] [*buffers*]
Alias: `b`
Prints the data in the specified system buffer according to *format*. If you omit *format*, the previous *format* is used. Valid formats include `decimal`, `octal`, `hex`, `character`, `byte`, `directory`, `inode`, `och`, `parcel`, `instruction`, and `write`. The `write` format creates a file in the current directory (see the FILES section) containing the buffer data.

`callout`
Aliases: `calls`, `call`, `c`, `timeout`, `time`, `tout`
Prints all entries in the callout table.

`ccall` [`-`|`--`|`---`|`----`] [`-g`] [*addr*]
Alias: `ag`
(DCE DFS only) Displays the list of Kernel RPC client call handle entries.

`-`, `--`, `---`, `----`
Specifies verbose level; displays additional information in the output. Increasing the number of dashes (the `-` character) increases the amount of information that is displayed.

`-g` Produces a command and address in `grep`(1) format that can be used to display a specific structure.

[*addr*] Limits the display to the data structure at the specified address. By default, all known data structures are displayed.

`cct` [`-`|`--`|`---`|`----`] [`-g`] [*addr*]
(DCE DFS only) Displays the list of Kernel RPC client connection table entries.

`-`, `--`, `---`, `----`
Specifies verbose level; displays additional information in the output. Increasing the number of dashes (the `-` character) increases the amount of information that is displayed.

`-g` Produces a command and address in `grep`(1) format that can be used to display a specific structure.

[*addr*] Limits the display to the data structure at the specified address. By default, all known data structures are displayed.

`ch` [`-`] *slot* | *address* [*slot* | *addresses* ...]
Formats the DFS client handle. If the `-` option is specified, a long-form report is generated that includes the NFS mount information structure and the remote procedure call (RPC) client structure.

`chtable` [`-`]
Aliases: `chtab`, `cht`
(DCE DFS only) Formats the DFS client handle table. If the `-` option is specified, a long-form report is generated that includes the NFS mount information structure and the RPC client structure.

`cku_private` *address* [*address* ...]
Aliases: `cku_priv`, `ckup`
Formats the NFS `cku_private` structures.

`clkar` Prints the `clock.c` 'ticks' distribution table.

`clusters`
Aliases: `cl`
Displays mainframe cluster information. This command is meaningful only when a guest is active.

`cm_conn` [`-`|`--`|`---`|`----`] [`-g`] [`-l`] [`-s`] [*addr*]
Alias: `cmc`
(DCE DFS only) Displays the Cache Manager connection list entries found in the Cache Manager server list.

`-`, `--`, `---`, `----`
Specifies verbose level; displays additional information in the output. Increasing the number of dashes (the `-` character) increases the amount of information that is displayed.

`-g` Produces a command and address in `grep`(1) format that can be used to display a specific structure.

`-l` Displays information about the lock structures contained in the structure(s) being displayed.

`-s` Displays information about Cache Manager servers as well as Cache Manager connection list entries.

[*addr*] Limits the display to the data structure at the specified address. By default, all known data structures are displayed.

`cm_serv` [`-`|`--`|`---`|`----`] [`-c`] [`-g`] [`-l`] [*addr*]
Alias: `cms`
(DCE DFS only) Displays information from the list of Cache Manager server structures (`cm_server`).

`-`, `--`, `---`, `----`
Specifies verbose level; displays additional information in the output. Increasing the number of dashes (the `-` character) increases the amount of information that is displayed.

`-c` Displays information about Cache Manager connection list entries as well as Cache Manager servers.

`-g` Produces a command and address in `grep`(1) format that can be used to display a specific structure.

`-l` Displays information about the lock structures contained in the structure(s) being displayed.

[*addr*] Limits the display to the data structure at the specified address. By default, all known data structures are displayed.

`codecmp`
Aliases: `ccmp`, `code`, `cc`
Compares the code sections of *namelist_filename* to the same addresses in the dump. This option does not work for a kernel that was compressed using the `kcompress`(8) command. You must first decompress the kernel.

`core` *proc_table_slot_number* [*filename*]
Aliases: `co`, `corefile`
Creates a file that contains the memory image of a given process. If you specify *filename*, it writes the image to that file; otherwise, `core` writes the image to a file named `core`. The core image cannot be created if the process is swapped.

`cpu` Aliases: `cp`, `cpus`
Displays information regarding CPU activity.

`dcache` [`-`|`--`|`---`|`----`] [`-g`] [`-f` *file*] [*addr*]
Alias: `dc`
(DCE DFS only) Displays information about Cache Manager `dcache` structures.

`-`, `--`, `---`, `----`
Specifies verbose level; displays additional information in the output. Increasing the number of dashes (the `-` character) increases the amount of information that is displayed.

`-g` Produces a command and address in `grep`(1) format that can be used to display a specific structure.

`-f` *file* Dump contents of the `CacheItems` file. The *file* argument specifies the path to this file. This option cannot be used with the *addr* argument.

[*addr*] Limits the display to the data structure at the specified address. By default, all known data structures are displayed. This argument cannot be used with the `-f` option.

`dfsmisc -z` [`-`|`--`|`---`|`----`] [`-g`] [*addr*]
Alias: `dfs`
(DCE DFS only) Displays all items in the `zlc` (zero link count) list. If the `-z` argument is not specified, the `dfsmisc` command does nothing.

`-z` Displays all items in the `zlc` list. This option is required.

`-`, `--`, `---`, `----`
Specifies verbose level; displays additional information in the output. Increasing the number of dashes (the `-` character) increases the amount of information that is displayed.

`-g` Produces a command and address in `grep`(1) format that can be used to display a specific structure.

[*addr*] Limits the display to the data structure at the specified address. By default, all known data structures are displayed.

`dfsstat` [`-c`] [`-g`] [`-p`] [`-r`]
(DCE DFS only) Displays DFS counters. If no options are specified, `dfsstat` repeats the previous option(s) or displays Protocol Exporter statistics (`-p` option) if no previous invocation was made.

`-c` Displays Cache Manager statistics

`-g` Produces a command and address in `grep`(1) format that can be used to display a specific structure.

`-p` Displays Protocol Exporter statistics. This option is the default.

`-r` Displays RPC (remote procedure call) statistics

`dhbuf` Displays device hash headers.

`dnlc` [`-s`]
Prints a formatted list of the directory name lookup cache. If the `-s` option is specified, the lookup cache usage statistics are printed.

`ds` [*list of addresses*]
Finds the closest data symbol to the specified addresses.

`err` Displays the error log.

`eslice` [`-`] [*major_numbers*]
Displays IOS model E slice information. By default, a list of active major slices is displayed. The `-` option adds minor slice information. Specify a list of major slice numbers to limit the display to those major slices. When major slice numbers are specified, minor slice information for those slices is also displayed.

`fid` [`-`|`--`|`---`|`----`] [`-f`] [`-g`] [`-h`] [`-l`] [`-t`] [*addr*]
(DCE DFS only) Displays entries from the Token Manager file ID hash table. With no options, `fid` executes as if the `-f` option was specified.

`-`, `--`, `---`, `----`
Specifies verbose level; displays additional information in the output. Increasing the number of dashes (the `-` character) increases the amount of information that is displayed.

`-f` Displays entries in the file ID hash table (`fidhash`). This option is the default.

`-g` Produces a command and address in `grep`(1) format that can be used to display a specific structure.

`-h` Displays `fshost` (file system host) information.

`-l` Displays information about the lock structures contained in the structure(s) being displayed.

`-t` Displays Token Manager (`tkm`) information.

[*addr*] Limits the display to the data structure at the specified address. By default, all known data structures are displayed.

`file` [*file_table_entries*]
Aliases: `files`, `f`
Formats the specified file table.

`flock` [`-s`]
Alias: `fl`
Prints file locking information; `-s` prints a summary.

`fshost` [`-`|`--`|`---`|`----`] [`-g`] [`-l`] [*addr*]
(DCE DFS only) Displays information about all file system hosts (`fshosts`) found in either the `fshs_priHostID` table or the `fshs_secHostID` table.

`-`, `--`, `---`, `----`
Specifies verbose level; displays additional information in the output. Increasing the number of dashes (the `-` character) increases the amount of information that is displayed.

`-g` Produces a command and address in `grep`(1) format that can be used to display a specific structure.

`-l` Displays information about the lock structures contained in the structure(s) being displayed.

[*addr*] Limits the display to the data structure at the specified address. By default, all known data structures are displayed.

`gch` [*kernel_id*]
Formats and prints the guest channel (`gch`) table for all guests or for the guest with the specified kernel ID.

`gcom` Formats and prints the guest communications (`gcom`) table.

`gcx` Formats and prints the guest context (`gcx`) table.

`gpf` Formats and prints the guest performance (`gpf`) table for all guests.

`gpi` Formats and prints the guest pseudo interrupt (`gpi`) table.

`gpq` [*kernel_id*]
Formats and prints the guest packet queue (`gpq`) table for all guests or for the guest with the specified kernel ID.

`grt` Formats and prints the guest global resource (`grt`) table.

`gsn` [*count*]
Formats and prints the guest snapshot buffer and dumps the specified number of entries (32 by default) from the guest snapshot trace buffer in reverse chronological order.

`guest` [*kernel_name*]
Sets the dump memory bias to the specified guest system (by default, the first active guest system). All subsequent `crash` commands are relevant to the new memory bias. If a `crash` binary (associated by name) is available, the user will be asked if they wish to use that binary to examine the guest portion of the dump, as shown in the following example:

```
# ./crash dump unicos
> guest
Use ./crash_galeg1 (y|n)? > y
 Running ./crash_galeg1 -s -g galeg1 dump ./unicos_galeg1
>
```

`hblk` [*hblk_slot_numbers* | *addresses*]
Displays information about hash blocks (`hblks`) in the buffer cache. For example, `hblk 1 3 7` displays information about the blocks in slots 1, 3, and 7 of the hash blocks (`hblk`) table. Any *address* specified must be decimal.

`hbuf` [`-`] [`-c`] [`-l`] [`-s`] [*hbuf_slot_number* | *addresses*]
Displays information about hash header entries in the buffer cache. Any *address* specified must be decimal.

`-` (same as `-l` option) Displays chains of `hblks` hashed to headers in addition to the default display.

`-c` Displays only chains of `hblks` hashed to headers.

`-l` Displays chains of `hblks` hashed to headers in addition to the default display.

`-s` Displays hash header usage statistics.

`help` [*command*]
Aliases: `?`, `h`
If you specify `help` alone, prints a synopsis of all of the commands. If you specify *command*, `help` provides a more in-depth description of the specified command.

`host` Sets the dump memory bias to host memory (zero (0)). All subsequent `crash` commands are relevant to the new memory bias.

`inode` [`-`] [*inode_table_entries*]
Aliases: `ino`, `i`
The `inode` command has been disabled. Use the `vnode` command or the `nc1inode` command.

`iobuf` [`-`] [*iobuf_table_entries*]
Alias: `ddutab`
Prints the Physical Device table.

`jobs` [`-`] [*job_table_entries*]
Alias: `jtab`, `sess`
Prints the System Job (Session) Table. Without options, `jobs` generates a one-line description of all jobs (sessions). The `-` option produces a long listing of the `session` structure contents. Additional fields display the IPC usage: messages, semaphores, and shared memory (CRAY T90 systems only).

`kbds` Displays Kernel-Based Device Simulator information.

`kfp` [`-a`|`-k`|`-u`] [*argument_pointer stack_frame_pointer stack_base*]

Aliases: `r5`, `fp`

Prints the program's idea of the start of the current stack frame if you do not specify an argument, or sets the frame pointer to the supplied values. Register B01 contains the argument pointer, and B02 contains the stack pointer when a process is in the kernel. The `registers` command can be used to obtain the values of B01 and B02 from a dump. If *stack_base* is not present, the kernel stack is assumed. The `user` command can be used to get the address of the saved stack in the user structure and B01 and B02 for the three register save areas. Usually, the first save area contains the B01 and B02 registers that should be used. (The `stack` command defaults to these values.)

`-a` Specifies an alternate LAT table to use for address mapping. Selects absolute, one-for-one mapping.

`-k` Selects the kernel LAT table to use for address mapping.

`-u` Selects the user LAT table to use for address mapping.

`kftrace` [`1`] [`a`] [`c`] [`f`] [`k`] [`n`] [`t`]

Displays kernel flowtrace information. By default, `kftrace` sorts the output inversely by total time in function (the `t` argument).

`1` Excludes functions called only once

`a` Sorts output by average time in function

`c` Displays times in microseconds (default is raw clock periods)

`f` Sorts output inversely by calling frequency

`k` Stores information internally, so the same data can be displayed several ways

`n` Sorts output by name

`t` Sorts output inversely by total time in function (default)

`lat -d` [*addressing mode*]

Redefines the default LAT table and default mode. With no arguments, the `-d` option displays the default LAT table and mode.

addressing
Specifies type of addressing: `absolute` (all logical-to-physical address mapping is one-to-one), `kernel`, or `user`.

mode Specifies mode as any combination of `r` (read), `w` (write), or `x` (execute).

`lat` [`-k`|`-u`]

Displays the current kernel (`-k`) and/or user (`-u`) LAT tables.

`lat -k`|`-u` [`-t` *memory_type*] [`-m` *machine_type*] *XP_addr*

Defines the current kernel (`-k`) or user (`-u`) LAT table based on the specified exchange package (*XP_addr*).

`-m` *machine_type*
Selects the machine type to help specify the exchange package.

`-t` *memory_type*
Selects memory type to help specify the exchange package.

`lat -k|-u` [*entry*: *mode ba la pb*]
Redefines the current kernel (`-k`) or user (`-u`) LAT table.

entry Specifies the LAT entry (such as `lat0` or `lat1`).

mode Specifies the mode as any combination of `r` (read), `w` (write), and `x` (execute).

ba Specifies the logical base address.

la Specifies the logical limit address.

pb Specifies the physical bias, which equals the physical base address minus the logical base address.

`lat -k|-u` [*entry*: *clear*]
Clears the current kernel (`-k`) or user (`-u`) LAT table entry. The *entry* argument specifies the LAT entry (such as `lat0` or `lat1`).

`ldch`
Alias: `ldcache`
Prints a summary of the logical device cache buffers.

`ldchage` [`-`]
Displays a summary of logical device cache (ldcache) aging information. The `-` argument displays aging information for each cache block.

`ldmap` [*ldmap_table_entries*]
Alias: `ddmaps`
Prints the logical device maps.

`leb` Displays the kernel multi-threading lock violations logged in the lockrule error buffer.

`lnode` [`-`] [*lnode_table_entries*]
Aliases: `lno`, `l`
Prints the lnode table entries.

`loadem`
Forces a process to appear as if in core. This is useful when looking at a process that has been swapped, but its memory space has not yet been reused.

`map` [`-`] [`l`] [*map_names* | *map_addresses*]
Displays the specified `map` structures. Values for *map_names* are `bmrmap`, `coremap`, `execmap`, `mcache`, `sdsmap`, `swapmap`, and `all`. If *map_addresses* is specified, the data beginning at that address are displayed as if it were a `map`. By default, all maps are displayed.

`-` Includes a list of allocated and free areas.

`l` Includes an octal dump of the map words.

`mdw` Prints the memory descriptor words from the dump header.

`mem` [`-`[*any_character*]]
Prints a map of processes and shared text segments in memory. On CRAY T90 systems, this command also displays shared memory segments. The `-` option also displays the `sched` control structure. If the `-` option is followed by any character (for example, `-z`) the `Nic` (nice value) and `Pri` (priority) files in the output are replaced by `Skip`, which represents the line number within `sched.c` where the process was skipped over for selection.

`mme` [`-a`] [`-r`] [*CPU_numbers*]
(C90 and T90 systems only) Displays memory error status save area entries for the specified CPUs. By default, non-null entries for all CPUs are displayed.

`-a` Forces display of null entries.

`-r` Specifies raw mode; displays the `PORT` and `READ MODE` or `DESTINATION` fields as octal values, rather than as a descriptive (mnemonic) interpretation. In addition, the `MEMORY ERROR ADDRESS` field is displayed as a single octal value, rather than being separated into section, subsection, bank group, and bank (C90 systems only).

`mount` [`-`] [`-`] [*slot_number*]
Aliases: `mnt`, `m`
Formats the mount table. If the `-` option is specified, a long-form report is generated that includes the `vfs` entry. The `--` option generates a long-form report for every table entry.

`msgq` [`-`]
Displays the IPC message queue structures. With no options, information on the `msqid_ds` structures is displayed. The `-` option displays additional information.

`mthold` [`-`]
Alias: `mth`
Displays information for CPUs that are currently holding on a kernel multi-threading lock and for CPUs that have panicked trying to unlock an already unlocked multi-threading lock. The `-` option displays additional information about each lock.

`mtlocks` [`-`] [`sem`*nn* | `SEM`*nn* | *address* | *symbol*]
Alias: `mtlock`, `mtl`, `mt`
Displays information for kernel multi-threading locks. By default, information about a fixed list of locks is displayed. The `-` option displays additional information about each lock. Individual `SEMLOCKS` can be displayed using the option `sem`*nn*; *nn* specifies the controlling hardware semaphore number (decimal). Individual `MEMLOCKS` can be displayed using their addresses using the *address* option. In the `mtlocks` display, an `S` character in the `Status` field means that the controlling semaphore bit for a `SEMLOCK` was set; an `L` character means that the memory lock word was set.

`mux` Displays information in the MIOP tables, the tables controlling I/O to Model-E clusters, and the associated channel tables.

`nc1inode` [`-`] [`-`] [*slot_number*]
Alias: `nc`
Formats the `nc1inode` table. If the `-` option is specified, a long-form report is generated that includes the vnode and the attribute structure. The `--` option generates a long-form report for every table entry.

`nfsmi` *address* [*address...*]
Prints the specified NFS mount info structure(s).

`nm` *list of symbols*
Prints symbol value and type as found in *namelist_filename*.

`noprint`
Aliases: `nopr`, `np`
Turns off output to the terminal.

`od` [`-a`|`-k`|`-u`] [`-m` *mode*] [`-t` *memory_type*] [*address*|*symbol*] [*count*] [*format*]
Aliases: `dump`, `rd`
Dumps *count* data values starting at *address* (or *symbol*) according to *format*. Use the `-a`, `-k`, and `-u` options to specify other than the default LAT table for address mapping.

`-a` Specifies absolute (one-to-one) address mapping of the LAT table.

`-k` Specifies the kernel LAT table.

`-m` *mode*
Specifies the address mode as any combination of `r` (read), `w` (write), or `x` (execute).

`-t` *memory_type*
Displays the data from those portions of *core_filename* defined by *memory_type* memory descriptor words (`mdws`). Any valid `mdw` may be specified (use the `mdw` subcommand to display the mdws in *core_filename*). The initial default is `mem`. Once *memory_type* is specified, it remains the default for subsequent `od` subcommands until overridden. This option cannot be used on a running system.

`-u` Specifies the user LAT table.

address Specifies address; octal word addresses by default. *address* can be followed by one of the following:

`p` Denotes parcel address.

`a`,`b`,`c`,`d` Denotes a word address plus parcel offset.

`B` Denotes byte address.

format Specifies format. Allowed formats are `och`, `octal`, `decimal`, `hex`, `character`, `byte`, `parcel`, and `instruction` (abbreviated as `I` or `i`). The default for *format* is `och` (`octal` plus `character`).

`packet` *queue_name* | *octal addresses*
Aliases: `pack`, `pkt`, `pk`
Formats the packet queues. The possible choices for *queue_name* are: `ios`, `iosin`, `iosout`, and `ssd`. Queues are dumped in reverse order, most recent first, with unprocessed packets followed by old packets. When displaying a running system, the information may be inconsistent or erroneous. If a list of addresses is specified, `crash` tries to format what it finds at each address as an IOS packet.

`pbuf` [*pbuf_header_table_slot_number*]
Aliases: `pbufhdr`, `phdr`
Formats the system physical buffer headers.

`pddtab` [`-l`] [`-`] [*slot_number*]
Alias: `pdd`
(Cray PVP systems with IOS model E only) Prints the contents of `pddtab` for defined devices.

`-` Provides a long listing of contents for all devices.

`-l` Provides a very long listing for all devices.

slot_number Provides details for specified slot.

For Cray PVP systems with IOS model E, this command formats and displays the contents of the `pddtab` table for the physical disks in the system dumped.

`pktdi` [`-d` | `-i`] | [*device* [*device* ...]]
Formats the IPI-3/IPI packet driver traces. If no options are specified, a list of the devices and associated trace buffer pointers is displayed.

`-` Formats traces for all devices.

`-d` Formats traces for each IPI-3 device.

`-i` Formats traces for all IOP devices and the`reqt` device.

device Formats traces for the specified device or devices.

`pktdk` [`-d` | `-i`] | [*device* [*device* ...]]
Formats the IPI-3/HIPPI packet driver traces. If no options are specified, a list of the devices and associated trace buffer pointers is displayed.

`-` Formats traces for all devices.

`-d` Formats traces for each IPI-3 device.

`-i` Formats traces for all IOP devices and the`reqt` device.

device Formats traces for the specified device or devices.

`pp` Prints the process management hash tables and active process links.

`print` Alias: `pr`
Turns on output to the terminal.

`prnode` [`-`] [`-`] [*slot_number*]

Alias: `prn`

Formats the `prnode` table. If the `-` option is specified, a long-form report is generated that includes the vnode and the attribute structure. The `--` option generates a long-form report for each table entry.

`proc` [`- | -r | -l | -w`] [*process_table_entries*]

Aliases: `ps`, `p`

Formats the process table. One of the following options can be specified:

`-` Generates a longer listing.

`-r` Displays only processes that can be run.

`-l` Displays additional information about each process.

`-w` Displays the event field symbolically, if possible.

`pws` [*CPU_numbers*]

Displays the `pws` structures for the specified CPUs. If *CPU_numbers* is not specified, `pws` displays the `pws` structures for all CPUs. Included in the `pws` structure are the addresses of the `unix`, `user`, and `diag` exchange packages for the CPU.

`redirect` [`?`] [`+`] [*file*]

Aliases: `redir`, `>`, `>>`

Sends a copy of all output to the specified file. Specify `redirect` with no *file* argument to stop sending output to the file. `redirect ?` prints the current state of redirection. `redirect +` or `>>` appends the output to the specified file, *file*. There must be a space between `>` or `>>` and the *file*.

`registers` *CPU registers*

Aliases: `register`, `regs`, `reg`

Prints the B, T, and V registers from a dump. This command works only from a saved core file; it does not work on a running system. The register list can be individual registers, such as B01, V123, T23, or it can be a register type followed by an asterisk, as shown in the following example:

```
reg cpu3 B01 B02 V*
```

If you do not specify a CPU number, the default is the last legal CPU used on a `registers` command.

`resinfo` [*resinfo_table_entries*]

Prints the `resinfo` structures.

`rnode` [`-`] *slot* | *address* [*slot* | *address* ...]

Alias: `rn`

Formats an NFS rnode. If the `-` option is specified, a long-form report is produced that includes the vnode and the attribute structure. If *slot* is specified, a report is produced for that rnode. If *address* is specified, it is interpreted as an rnode address.

`rnodetab` [`-`]

Alias: `rnt`

Formats the NFS rnode table. If the `-` option is specified, a long-form report is generated that includes the vnode and attribute structure for each rnode.

`rpe` [`-a`] [`-r`] [`-s`] [`-u`] [*CPU_numbers*]

(Y-MP, C90 and T90 systems only) Displays the register parity error (`rpe`) status save area entries for the specified CPUs. (For each configured CPU, there are two slots in the `rpe` save area; one for the status of the most recent `rpe` that occurred while in system mode, and one for the most recent `rpe` in user mode.) By default, `rpe` displays the non-null system and user mode entries for all CPUs.

`-a` Forces display of null entries.

`-r` Specifies raw mode; displays the `RPE ERROR BITS` field in octal. By default, the descriptive (mnemonic) interpretation is displayed.

`-s` Displays system mode entries.

`-u` Displays user mode entries.

`rslogdump` > *filename*

Alias: `rslog`

(Secure kernel with buffered security logs records only; that is, records queued for read by the security log daemon) Dumps to *filename* all security log records that have not been buffered by the security log daemon. The data is dumped in a raw format; therefore, redirection is necessary. The resulting file can then be examined by using the `reduce`(8) command.

`scache` [`-`|`--`|`---`|`----`] [`-a`] [`-g`] [`-l`] [`-v`] [*addr*]

Alias: `sc`

(DCE DFS only) Displays Cache Manager `scache` entries.

`-`, `--`, `---`, `----`

Specifies verbose level; displays additional information in the output. Increasing the number of dashes (the `-` character) increases the amount of information that is displayed.

`-a` Displays the `vattr` attribute structure using the format of the `nc1inode` subcommand.

`-g` Produces a command and address in `grep`(1) format that can be used to display a specific structure.

`-l` Displays information about the lock structures contained in the structure(s) being displayed.

`-v` Displays the vnode using the format of the `nc1inode` subcommand.

[*addr*] Limits the display to the data structure at the specified address. By default, all known data structures are displayed.

`sct` [`-`|`--`|`---`|`----`] [`-g`] [*addr*]

(DCE DFS only) Displays information about Kernel RPC `sct` structures.

-, --, ---, ----
Specifies verbose level; displays additional information in the output. Increasing the number of dashes (the - character) increases the amount of information that is displayed.

-g Produces a command and address in grep(1) format that can be used to display a specific structure.

[*addr*] Limits the display to the data structure at the specified address. By default, all known data structures are displayed.

sema [-]
Displays the IPC semaphore structures. With no options, information on the semid_ds structures is displayed. The - option displays additional information.

semundo [-]
Alias: semu
Displays the IPC semaphore undo structures. With no options, this command displays information on the undo structures and the free and in-use chains. The - option displays information on the semaphore undo values.

sess [-] [*job_table_entries*]
Alias: jtab, jobs
Displays the System Job (Session) Table. Without options, sess generates a one-line description of all sessions (jobs). The - option produces a long listing of the session structure contents. Additional fields display the IPC usage: messages, semaphores, and shared memory (CRAY T90 systems only).

shm [-]
(CRAY T90 systems only) Displays information about the shared memory segments that exist on the system. The - option displays the configuration values.

shrc Displays the fair-share scheduler shrconst structure. This information is also available with the shradmin -v command; see the shradmin(8) man page for more information.

siginfo *proc_table_entries*
Alias: sig
Prints signal information for the specified processes.

sizeof *structure_names*
Prints the size of the specified structures.

slice [-l] [-m] [-p] [-r] [-s] [-S] [*minor number*]
Alias: sli
(Cray PVP systems with IOS model E only) Prints eslice definition for defined devices.

-l Provides dev_ldd (logical devices).

-m Provides dev_mdd (mirrored devices).

-p Provides dev_pdd (physical devices).

`-r` Provides `dev_rdd` (RAM devices).

`-s` Provides `dev_sdd` (striped devices).

`-S` Provides `dev_ssdd` (SSD).

minor number
Provides details for the specified *minor number*.

Example: `slice -l 3` prints `dev_ldd` minor number 3.

This command (without options) provides a long listing of the defined slices (`eslice` for all the recognized types of devices in the system (physical: `dev_pdd`, logical: `dev_pdd`, mirrored: `dev_mdd`, SSD: `dev_ssdd`, and striped: `dev_sdd`). The information printed is for IOS-model-E based systems because they are the only systems that contain defined `eslice` structures. Physical slices are printed as stand-alone definitions whereas the logical slices are printed with the physical slices that define the device space.

`slogstat`
Alias: `slog`
(Secure kernel with logging enabled only) Formats the `slg` structure. Displays various information about the security log status (such as the size, status, location, and offset).

`slot` *table_name address*
Gives the slot number in the given table, *table_name*, for the specified *address*. You can then use the slot number to print the table entry. For example, if word address 112457 is in `proc` table entry 25, the following returns slot number 25:

```
slot proc 112457
```

The following command formats this `proc` table entry:

```
proc 25
```

`slr` Displays the Shared Lock Region on systems running the Shared File System (SFS).

`smp` Displays SMP-1 and SMP-2 hardware semaphore device tables.

`ssd` Displays SSD configuration and statistics.

`ssddtab`
Alias: `ssdd`
Displays the SSD device tables.

`stack` [`-`] *process_table_entries*
Aliases: `stk`, `s`, `kernel`, `k`
Formats a dump of the kernel stack of a process from the `ublock` of the process. If the process was executing at the time of the dump, the stack pointers are obtained automatically from the saved registers for that CPU. If the process was not in the kernel, the stack trace may be garbage. The `-` option produces more extensive formatting of each stack frame. For explicit control of the starting stack frame, use the `registers` command to get the B01 and B02 registers, the `kfp` command to set these values, and the `trace` command with the `-r` option to print the stack. When looking at a dump taken from a different type of machine than the current one, the line numbers printed in the trace may be wrong.

`stat` Prints certain statistics found in the dump. These include the panic string (if a panic occurred), time of deadstart, and the CPU and process that were last in the kernel.

`swapmap` [`-`[*any_character*]]
Prints the swap file allocation map. The `-` option also displays the `sched` control structure. If the `-` option is followed by any character (for example, `-z`) the `Nic` (nice value) and `Pri` (priority) files in the output are replaced by `Skip`, which represents the line number within `sched.c` where the process was skipped over for selection.

`swapper`
Prints the `sched control structure.`

`swapq` [`-`[*any_character*]]
Prints a list of swapped processes that are eligible for swap-in. The `-` option also displays the `sched` control structure. If the `-` option is followed by any character (for example, `-z`) the `Nic` (nice value) and `Pri` (priority) files in the output are replaced by `Skip`, which represents the line number within `sched.c` where the process was skipped over for selection.

`svc_data` *address* [*address* ...]
Aliases: `svc_d`, `svcd`
Formats the NFS `svc_data` structures.

`svc_xprt` *address* [*address* ...]
Aliases: `svc_x`, `svcx`
Formats the RPC `SVCXPRT` structures.

`sysent`
Prints the system call timings found in the `sysent` structure within the kernel.

`sysint`
Prints the system interrupt timings found in the `sysint` structure within the kernel.

`tabinfo` *address*
Aliases: `tabinit`, `tab`
Sets for `crash` where the `tabinfo` structure resides to address.

`text` [*text_table_entries*]
Aliases: `txt`, `x`
Formats the text table.

`tkc` [`-`|`--`|`---`|`----`] [`-g`] [`-l`] [*addr*]
(DCE DFS only) Displays information about Token Cache (`tkc`) structures.

`-`, `--`, `---`, `----`
Specifies verbose level; displays additional information in the output. Increasing the number of dashes (the `-` character) increases the amount of information that is displayed.

`-g` Produces a command and address in `grep`(1) format that can be used to display a specific structure.

`-l` Displays information about the lock structures contained in the structure(s) being displayed.

[*addr*] Limits the display to the data structure at the specified address. By default, all known data structures are displayed.

`tkm` [`-`|`--`|`---`|`----`] [`-f`] [`-g`] [`-h`] [`-l`] [`-t`] [*addr*]
(DCE DFS only) Displays information from the Token Manager (`tkm`) list. If no options are specified, `tkm` behaves as if the `-t` option is selected.

`-`, `--`, `---`, `----`
Specifies verbose level; displays additional information in the output. Increasing the number of dashes (the `-` character) increases the amount of information that is displayed.

`-f` Displays entries in the Token Manager file ID hash table (`fidhash`).

`-g` Produces a command and address in `grep`(1) format that can be used to display a specific structure.

`-h` Displays `fshost` (file system host) information.

`-l` Displays information about the lock structures contained in the structure(s) being displayed.

`-t` Displays Token Manager (`tkm`) information. This is the default option.

[*addr*] Limits the display to the data structure at the specified address. By default, all known data structures are displayed.

`tkset` [`-`|`--`|`---`|`----`] [`-g`] [`-l`] [*addr*]
(DCE DFS only) Displays information about Token Sets (`tkset`).

`-`, `--`, `---`, `----`
Specifies verbose level; displays additional information in the output. Increasing the number of dashes (the `-` character) increases the amount of information that is displayed.

`-g` Produces a command and address in `grep`(1) format that can be used to display a specific structure.

`-l` Displays information about the lock structures contained in the structure(s) being displayed.

[*addr*] Limits the display to the data structure at the specified address. By default, all known data structures are displayed.

`tpq` [*addr*]

(DCE DFS only) Displays information about Thread Pool Queues (`tpq`).

[*addr*] Specifies the address returned from the `tpq_Init()` routine.

The following thread handles are available:

- `cm_auxThreadPoolHandle`
- `cm_threadPoolHandle`
- `tkm_threadPoolHandle`

The following example shows the use of the `od` subcommand to obtain the address for the thread handle `cm_threadPoolHandle`. This address is then used as the *addr* argument for `tpq`.

```
> od cm_threadPoolHandle
2645314: 000000000000117661644
> tpq 011766164S
```

`tps` [*device1*] [*device2*] . . .

Prints tape device structures. With no arguments, `tps` displays the tape I/O structures for all tape devices in the system. When a device name is specified, `tps` displays the tape structures associated with that device. Using a `-` character instead of *device1* prints out tape structures for all tape devices in the system.

`tpt` [*device1*] [*device2*] . . .

Prints kernel level tape device traces. `tpt` called without any arguments prints out a table containing the device name (as seen in the `tpstat`(1) display), index (physical device name), and the start, middle and end trace pointers for each device in the the tape table. `tpt` called with a device name prints out traces for that device. `tpt` called with `-` instead of *device1* dumps out traces for all tape devices in the system.

`trace` [`-`] [`-r`]

Aliases: `t`, `ytrace`, `ytra`, `ytr`, `yt`

Generates a kernel stack trace. The trace begins at the saved stack frame pointer in `kfp`. A `-` followed by `-r` does more intricate formatting of the stack frames. The line numbers printed can be incorrect if the type of the machine from which the dump was taken does not agree with the type of the machine for which `crash` was built. The `ytrace` command forces the stack to be formatted in Y-mode.

`ts` *list of symbols*

Prints symbol value and type as found in *namelist_filename*.

`tty` [*type*] [`-`] [*tty_table_entries*]
Aliases: `term`, `pty`
Prints the `tty` structures. The *type* argument determines which structure will be used (such as `tty` or `pty`). The default for *type* is `tty`. However, after you have specified *type*, the last value specified is used. The `-` option prints additional information, including the `stty`(1) options for the given line.

`uio` [*slot_numbers*]
Displays the specified `uio` entries. By default, all `uio` entries are displayed.

`unloadem`
Forces a process to appear as if swapped.

`user` [*process_table_entries*]
Aliases: `uarea`, `u_area`, `u`
Prints the user structure of the specified process as determined by the information contained in the process table entry. If no entry number is specified, the information from the last executing process will be printed. Swapped processes produce an error message.

`utc` [`-`] [*major_numbers*]
Displays UTC (Universal Time Clock) device tables.

`utrace` [`-`] [`+`] [*count*]
Alias: `ut`
Dumps *count* entries from the kernel trace buffer in reverse chronological order (latest first). If you specify the `-`, `utrace` prints the address of each `utrace` entry. The `+` option causes the dump of the `utrace` entries to continue from the next trace buffer entry. The default *count* is 32 entries.

`var` Aliases: `tunables`, `tunable`, `tune`, `v`
Prints the system parameters that can be tuned.

`vfs` Formats the `vfs` and print in chain order.

`vhisp` Displays `vhisp` configuration and statistics.

`vnode` [*addr*]
Prints a single-line description of all vnodes. Vnodes are housed inside the dependent inodes. The slot numbers on this report refer to the dependent inode tables. If *addr* is specified, it is interpreted as a vnode address. If the address is a vnode, `crash` determines the file system and the dependent inode.

`volume` [`-`|`--`|`---`|`----`] [`-g`] [`-l`] [*addr*]
Alias: `vol`
(DCE DFS only) Displays information about known DFS volumes.

`-`, `--`, `---`, `----`
Specifies verbose level; displays additional information in the output. Increasing the number of dashes (the `-` character) increases the amount of information that is displayed.

`-g` Produces a command and address in `grep`(1) format that can be used to display a specific structure.

`-l` Displays information about the lock structures contained in the structure(s) being displayed.

[*addr*] Limits the display to the data structure at the specified address. By default, all known data structures are displayed.

`xp` [`-m` *machine_type*] [`-t` *memory_type*] *address*

Alias: `exch`

Formats the data at *address*, assuming that it is an exchange package.

`-m` *machine_type*

Format the data as if the hardware were *machine_type*. Values for *machine_type* are `y`, `ymp`, `y-ea`, `ymp-ea`, `craymp`, `crayymp`, `c90`, `crayc90`, `cray-j90`, `crayts`, `cray-ts`, `T90`, `t90`, `t16`, `t32`, `T90I`, `t90i`, `crayts-ieee`, `cray-ts-ieee`, `T90-ieee`, `t90-ieee`, `t4-ieee`, `t16-ieee`, and `t32-ieee`. The default is the machine type of the dumped system.

`-t` *memory_type*

Displays the data from those portions of *core_filename* defined by *memory_type* memory descriptor words (`mdws`). Any valid `mdw` may be specified; however, only the `mem` and `xp` types contain exchange packages (use the `mdw` subcommand to display the mdws in *core_filename*). The initial default is `mem`. Once *memory_type* is specified, it remains the default for subsequent `xp` subcommands until overridden. This option cannot be used on a running system.

`xpa` [`-m` *machine_type*]

Displays the exchange package areas for all active (host and guest) kernels. If the `-m` option is specified, the exchange packet is formatted for the hardware specified by *machine_type*. Values for *machine_type* are `y`, `ymp`, `y-ea`, `ymp-ea`, `craymp`, `crayymp`, `c90`, `crayc90`, `cray-j90`, `crayts`, `cray-ts`, `T90`, `t90`, `t16`, `t32`, `T90I`, `t90i`, `crayts-ieee`, `cray-ts-ieee`, `T90-ieee`, `t90-ieee`, `t4-ieee`, `t16-ieee`, and `t32-ieee`.

`xsinfo` [`-`]

Alias: `xsi`

Prints the Multiplexed (MPX) Scheduler information table. If the `-` option is used, data for swap partitions 0 through `SWAP_PARTS` (defined in `sys/swap.h`) is displayed. Otherwise, the display is limited to partitions 0 through `swapper.swp_nparts - 1`. The `-` option also forces the display when MPX scheduling is not configurable (that is, `swapper.swp_nparts = 1`).

There are built-in aliases for many of the *formats*, as well as those listed for the commands. Some of the aliases are as follows:

Format	Alias
byte	b
character	char, c
parcel	par, p
instruction	instr, ins, I
decimal	dec, e
directory	direct, dir, d
hexadecimal	hexadec, hex, h, x
inode	ino, i
longdec	ld, D
longoct	lo, O
octal	oct, o
pddtab	pdd
slice	sli
write	w

NOTES

Because most flags are abbreviated and have little meaning to an uninitiated user, a source listing of the system header files is useful while using `crash`.

BUGS

Stack tracing of the current process on a running system does not work.

FILES

`/usr/include/sys/*.h`	Header files for table and structure information
`/dev/mem`	Default system image file
`buf.#`	Files created that contain buffer data

SEE ALSO

`kcompress`(8), `mount`(8), `reduce`(8)

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

General UNICOS System Administration, Cray Research publication SG–2301

NAME

cron – Clock daemon

SYNOPSIS

/etc/cron [-m *limit*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The cron program executes commands at specified dates and times.

The cron program accepts the following option:

-m *limit* Sets the upper *limit* on number of jobs that can be run at once. (Allows you to specify a value for MAXRUN.) Default is 25.
NOTE: The limit on number of jobs that can be run at once is also subject to job queue limits (see queuedefs(5)).

Commands that are executed on a regular basis can be specified according to instructions found in crontab files; users can submit their own crontab file by using crontab(1). Commands that are executed only once can be submitted using the at(1) command. Because cron never exits, it should be executed only once. This is best done by running cron from the initialization process through the /etc/rc file.

The cron program examines crontab files and at(1) command files only during process initialization and when a file changes through the crontab(1) command. This reduces the overhead of checking for new or changed files at regularly scheduled intervals.

NOTES

There are factors that cause cron to not immediately process changes made to a crontab file through the crontab(1) command. These factors include long-running cron jobs and system overhead.

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 start the cron daemon.

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

MESSAGES

A history of all actions taken by `cron` is recorded in `/usr/lib/cron/log`.

FILES

`/usr/lib/cron`	Main `cron` directory
`/usr/lib/cron/log`	Accounting information
`/usr/lib/cron/queuedefs`	Definitions for all queues managed by `cron`
`/usr/spool/cron`	Spool area

SEE ALSO

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

`queuedefs`(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

csa – Overview of Cray Research system accounting

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The UNICOS operating system supports two accounting packages, Cray Research system accounting (CSA) and standard UNIX System V accounting. CSA is designed to meet the unique accounting requirements of Cray Research sites. It is a set of C programs and shell scripts that, like standard UNIX System V accounting, provides methods for collecting per-process resource usage data, recording connect sessions, monitoring disk usage, and charging fees to specific logins. CSA also provides the following facilities that are not available from the standard UNIX accounting package:

- Per-job accounting
- Device accounting
- Daemon accounting for monitoring the Network Queueing System (NQS) and the UNICOS tape subsystem
- Disk accounting by account ID
- Arbitrary accounting periods
- Flexible system bill unit (SBU) system
- One file containing all data for an accounting period
- Front-end formatting interfaces
- Offline archiving of accounting data

The UNICOS kernel performs process accounting. On termination of a process, one record per process is written to a file, usually /usr/adm/acct/day/pacct. At the completion of various daemon specific events, Network Queuing System (NQS) and tape daemons write daemon accounting records. The csabuild(8) command combines the data and generates a session record file, which is used as input by other CSA programs to generate reports, bills, and data for front-end systems.

FILES

/etc/csaboots	Captures system boot times.
/etc/udb	User validation file that contains user control limits; contains login name to user ID conversions.
/etc/wtmp	Contains login and logoff history information.
/usr/lib/acct	Contains most of the accounting commands listed in *UNICOS Resource Administration*, Cray Research publication SG–2302.

`/usr/lib/acct/day/pacct`	Contains current process accounting information.
`/usr/adm/acct/day/nqacct*`	CSA NQS accounting files.
`/usr/adm/acct/day/tpacct*`	CSA tape accounting files.

SEE ALSO

`acct`(8), `acctcms`(8), `acctdusg`(8), `accton`(8), `acctsh`(8), `acctwtmp`(8), `csaaddc`(8), `csaboots`(8), `csabuild`(8), `csacon`(8), `csaconvert`(8), `csacrep`(8), `csadrep`(8), `csaedit`(8), `csafef`(8), `csaibm`(8), `csajrep`(8), `csaline`(8), `csanqs`(8), `csapacct`(8), `csaperiod`(8), `csaperm`(8), `csarecy`(8), `csaswitch`(8), `csatape`(8), `csaverify`(8), `diskusg`(8), `dodisk`(8), `fwtmp`(8), `lastlogin`(8), `nulladm`(8), `shutacct`(8), `startup`(8), `turnacct`(8), `turndacct`(8)

`acctcom`(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

UNICOS Resource Administration, Cray Research publication SG–2302

NAME

csaaddc – Adds cacct records

SYNOPSIS

/usr/lib/acct/csaaddc [-a] [-o *ofile*] [-t] [-v] [[[-A] [-g]] [-j] [-u]] *ifiles*

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The csaaddc command adds cacct records and outputs cacct records. You also can generate cacct records with the csacon(8) and acctdisk(8) commands.

The csaaddc command accepts the following operand:

ifiles Specifies the files to be processed. The files are in cacct format.

Output Options

-a Outputs information in ASCII. By default, the output is written to a binary output file.

-o *ofile* Specifies the output file. By default, the output is written to stdout.

-t Totals all records into one record.

-v Sets verbose mode on. When you also specify -a, verbose output is written to stderr.

Consolidation Options

You can specify multiple consolidation options. These options should be the same as those used to create the input files with csacon(8). If you do not specify any consolidation options, the default is -Au.

-A Totals by account ID.

-g Totals by group ID. You must use this option with at least one of the other consolidation options, because not all records have a group ID.

-j Totals by job ID.

-u Totals by user ID.

ifiles This operand is a list of input files in cacct format. The file names are separated by spaces.

NOTES

The consolidation options used with csacon(8) to generate the input files should correspond to the consolidation options used with csaaddc. If they do not, the resulting data can be misleading and difficult to interpret.

Be aware that the `csacon -a` option corresponds to the `csaaddc -A` option.

You must consolidate disk data generated by `acctdisk -A` by using `csaaddc` with the `-A` option. `acctdisk -A` sets the *uid* and *jid* fields of all records to `0`; therefore, these records should not be consolidated by job ID or user ID.

EXAMPLES

The following example merges two input files created with `csacon`(8), using the `-a` and `-g` consolidation options. The output is written to file `outfile`.

```
csaaddc -A -g -o outfile cacct1 cacct2
```

SEE ALSO

`acctdisk`(8), `csacon`(8), `csacrep`(8), `diskusg`(8), `dodisk`(8)

UNICOS Resource Administration, Cray Research publication SG–2302

NAME

csaboots – Records system boot times for the accounting subsystem

SYNOPSIS

```
/etc/csaboots [-o csafile] [-u] [-v]
/etc/csaboots [-o csafile] [-U ufile] [-v]
```

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The csaboots command records system boot times for the accounting subsystem by writing a record of each boot time into the /etc/csainfo file. The default is to write the current time to the output file. rc (see brc(8)) invokes the csaboots command during system startup.

The csaboots command accepts the following options and arguments:

-o *csafile*	Changes the output file from /etc/csainfo to *csafile*.
-u	Writes all boot times found in /etc/utmp (utmp(5)) to the output file.
-U *ufile*	Writes all boot times found in *ufile* to the output file. *ufile* must be in utmp format.
-v	Sets verbose mode. Informational messages are written to stdout.

FILES

/etc/utmp	Records information about current system users
/etc/csainfo	Record of system boot times

SEE ALSO

utmp(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

csabuild – Generates a session record file

SYNOPSIS

/usr/lib/acct/csabuild [-a] [-A] [-C *ctmppath*] [-D *level*] [-i] [-n] [-N *nqspath*] [-o *nday*] [-P *pacctpath*] [-s *sessionfile*] [-S *segmentsize*] [-t] [-T *tapepath*] [-u *uptimepath*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The csabuild command organizes the accounting files from the following sources, into session records (session.h):

- Network Queuing System (NQS)
- Online tapes (tpdaemon(8))
- Kernel accounting
- Connect accounting

The daemon accounting programs use session records to generate reports, bills, and data for front-end systems.

An integer suffix is attached to the path name arguments; a 0 indicates recycled information, and 1 and higher indicates current data.

The csabuild command accepts several types of options: file name, performance, job ending, and debugging.

File Name Options

The csabuild command accepts the following file name options:

-C *ctmppath* Specifies connect time record path name for preprocessed data (output from csaline(8)). The default is /usr/adm/acct/work/Pctime.

-N *nqspath* Specifies NQS path name for the preprocessed data (output from csanqs(8)). The default is /usr/adm/acct/work/Pnqacct.

-P *pacctpath* Specifies pacct file path name. The default is /usr/adm/acct/work/Wpacct.

-s *sessionfile* Specifies session file name. The default is /tmp/Super-record. This is the csabuild output file.

-T *tapepath* Specifies tape daemon file path name. The default is /usr/adm/acct/work/Ptpacct.

-u *uptimepath* Specifies uptime path name. The default is /usr/adm/acct/work/Puptime.

Performance Options

The `csabuild` command accepts the following performance options:

`-S` *segmentsize* Changes default segment size. The default is 1000 jobs per segment. Debug level 3 displays the percentage of each segment used. If more jobs are run per day, increase this number. If less jobs are run per day, decrease this number.

`-t` Prints timing information for the two major phases of `csabuild`.

Job Ending Options

The `csabuild` command accepts the following job ending options:

`-a` Assumes crash option. The default operation is if a job does not have an associated end-of-job record, but the system was rebooted, the job is assumed to be terminated. With this option, these jobs are not marked as terminated.

`-o` *nday* Terminates the session if a session is older than *nday* days. NQS requests submitted more than *ndays* ago also are terminated. You can use this option to terminate old jobs that are known to be finished.

Debugging Options

The `csabuild` command accepts the following debugging options:

`-A` Abort option. If `csabuild` exits with an error, a core dump is generated.

`-D` *level* Controls messages printed during program execution. Level 1 is verbose, level 10 is not appropriate for any execution, except small test cases.

`-i` Ignores bad records. If `csabuild` runs into a record that it detects as bad, it can recover from the error by discarding the record and continuing to process input. When it discards a record, it prints a diagnostic message.

`-n` Suppresses the NQS sort and condense phase. This option prevents NQS jobs that span multiple system boots from being condensed into one job. (This function is intended only for error recovery.)

NOTES

The `pacct1` and `uptime1` files must exist, other files can be ignored; although without them, data on the associated daemons is not gathered. `csaverify`(8) can verify most of the input files. Generally, `csaedit`(8) and `csapacct`(8) can verify and repair bad input files.

BUGS

`csabuild` is limited by its input. Unless the data files are accurate, the sessions cannot be organized correctly.

FILES

`/usr/adm/acct/day` Directory that contains current unprocessed accounting data

SEE ALSO

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

UNICOS Resource Administration, Cray Research publication SG–2302

NAME

csacon – Condenses a session record file into a cacct file

SYNOPSIS

/usr/lib/acct/csacon [[[-a] [-j] [-u]] [-g]] [-s *sessionfile*] [-v] [-A] [-D *level*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The csacon command condenses the information from a session record file into cacct format. You can use the csacrep(8) command to generate ASCII reports from the consolidated file. csacon accepts several types of options: consolidation, input, and output options.

Consolidation Options

The csacon command accepts the following consolidation options. You can specify multiple consolidation options. If you do not specify any consolidation options, the default is to use -au.

-a Consolidates session records by using the account ID as a key.

-A Consolidates all jobs, including those that have not completed. By default, only jobs that have completed are consolidated.

-D *level* Sets the debugging level. Level 1 is slightly verbose; level 10 is very verbose. Debug output is written to standard error. By default, debugging is turned off.

-g Consolidates session records by using the group ID as a key. Because not all records have a group ID, you must use this option with at least one of the following consolidation options: -a, -j, or -u.

-j Consolidates session records by using the job ID as a key.

-u Consolidates session records by using the user ID.

Input Option

The csacon command accepts the following input option:

-s *sessionfile*
Specifies the name of the session record file, which is the input file. csabuild(8) created the file. The default is /tmp/Super-record.

Output Option

The csacon command accepts the following output option:

-v Sets verbose mode on. Verbose output is written to standard error.

EXAMPLES

The following example consolidates all records in the session record file `srec`. The session records are condensed by the three-tuple account ID, job ID, and user ID. Output is written to the `cacct` file.

```
csacon -A -a -j -u -s srec > cacct
```

SEE ALSO

`csaaddc`(8), `csabuild`(8), `csacrep`(8)

UNICOS Resource Administration, Cray Research publication SG–2302

NAME

csaconvert – Converts UNICOS 8.0, 8.3, 9.0, 9.1, 9.2, and 9.3 accounting file(s) to UNICOS 10.0 format

SYNOPSIS

/usr/lib/acct/csaconvert [-L *level*] [-m] [-v] [-N *nqacctfile*] [-o *outfile*]

/usr/lib/acct/csaconvert [-L *level*] [-m] [-v] [-a *tacctfile*] [-o *outfile*]

/usr/lib/acct/csaconvert [-L *level*] [-m] [-v] [-c *cacctfile*] [-o *outfile*]

/usr/lib/acct/csaconvert [-L *level*] [-m] [-v] [-n *Pnacctfile*] [-o *outfile*]

/usr/lib/acct/csaconvert [-L *level*] [-m] [-v] [-p *pacctfile*] [-o *outfile*]

/usr/lib/acct/csaconvert [-L *level*] [-m] [-v] [-s *cmsfile*] [-o *outfile*]

/usr/lib/acct/csaconvert [-L *level*] [-m] [-v] [-t *tpacctfile*] [-o *outfile*]

/usr/lib/acct/csaconvert [-L *level*] [-m] [-v] [-o *outfile*]

/usr/lib/acct/csaconvert [-L *level*] [-m] [-v] [-x *tacctfile*] [-o *outfile*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The csaconvert command converts UNICOS 8.0, 8.3, 9.0, 9.1, 9.2, and 9.3 accounting files to UNICOS 10.0 format. You can specify either a file name or a path name for the input and the output. If you specify a path name, you must use the -m option. For files that already have been converted, the csaconvert command will display their name(s); it will not perform the conversion. csaconvert accepts several types of options: processing, input, and output options.

Processing Options

The csaconvert command accepts the following processing options:

-L *level* Specifies the UNICOS release level under which the input files were generated. The default release level is UNICOS 8.0.

-m Specifies that the input and output file names are path names. A numeric suffix, starting at either 0 or 1, is automatically appended to the file name. If you omit the -o option, all output is written to the stdout file (standard output).

-v Verifies the file revision level. (No conversion is done.)

Input Options

The csaconvert command accepts the following input options. Only one input option may be specified. By default, the input is the pacct accounting file Wpacct1 (-p).

-N *nqacctfile* Specifies a raw NQS file created by the NQS daemon.

-a *tacctfile* Specifies a tacct file created by the acctprc2(8) command.

-c *cacctfile* Specifies a cacct file created by the csaaddc(8) or csacon(8) command.

-n *Pnacctfile* Specifies a preprocessed NQS file created by the csanqs(8) command.

-p *pacctfile* Specifies a pacct accounting file. This is the default input with Wpacct1 as the file name.

-s *cmsfile* Specifies a cms file created by the acctcms(8) command.

-t *tpacctfile* Specifies a tape accounting file created by the tape daemon.

-x *tacctfile* Specifies a tacct file for conversion to cacct format. This capability is necessary if you are converting from UNIX System V accounting to Cray Research system accounting.

Output Option

The csaconvert command accepts the following output option:

-o *outfile* Specifies the output file. The default is stdout.

NOTES

All UNICOS 10.0 accounting tools are able to process accounting data generated on systems running UNICOS 8.0, 8.3, 9.0, 9.1, 9.2, and 9.3. As needed, the data is converted automatically to UNICOS 10.0 format. Since the UNICOS 9.0 release, you are no longer required to run the csaconvert command to convert the prior accounting data to the current release format.

However, if you access the prior accounting data on a regular basis, for performance reasons you should convert the data once using the csaconvert command. This allows you to avoid the overhead of repeatedly converting the data automatically. In this instance, explicit conversion is preferred.

EXAMPLES

Example 1: The following example converts a UNICOS 8.0 cacct file to UNICOS 10.0 format. The output is written to cacct100.

```
/usr/lib/acct/csaconvert -c cacct -o cacct100
```

Example 2: The following example converts a UNICOS 9.3 cacct file to UNICOS 10.0 format. The output is written to cacct100.

```
/usr/lib/acct/csaconvert -L 9.3 -c cacct -o cacct100
```

Example 3: The following example converts UNICOS 8.0 `pacct` files named `Wpacct0`, `Wpacct1`, `Wpacct2`, and so on, found in `/usr/adm/acct/sum/data/0413/1800`. The output is written to the `Npacct0`, `Npacct1`, and `Npacct2` files, and so on.

```
/usr/lib/acct/csaconvert -p /usr/adm/acct/sum/data/0413/1800 -o Npacct -m
```

Example 4: The following example converts a UNICOS 9.3 System V `tacct` file to UNICOS 10.0 `cacct` format:

```
/usr/lib/acct/csaconvert -L 9.3 -x tacct -o cacct
```

SEE ALSO

`acctprc`(8), `csaaddc`(8), `csabuild`(8), `csacon`(8)

UNICOS Resource Administration, Cray Research publication SG–2302

NAME

csacrep – Reports on consolidated accounting data

SYNOPSIS

/usr/lib/acct/csacrep [-a | -u] [-b] [-c] [-d] [-f] [-g] [-h] [-j] [-m] [-n] [-w] [-x] [-y] [-C] [-J] [-M]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The csacrep command generates reports from data in cacct format, such as output from the csacon(8) command. The default output reports the account name, the user ID, and the login name, and it suppresses the headers.

The csacrep command accepts the following two types of options: sorting and printing.

Sorting Options

The csacrep command accepts the following two sorting options. You cannot use the -a and -u options together.

-a Sorts output first by the account ID and then by the secondary key of the user ID. By default, csacrep does not sort. You cannot use this option with the -u option.

-u Sorts output by user ID and then by the secondary key of the account ID. By default, csacrep does not sort. You cannot use this option with the -a option.

Printing Options

The csacrep command accepts the following printing options.

-b Reports SBU data.

-c Reports CPU time memory integral and connect time in seconds.

-d Reports the cumulative online and offline disk usage and the number of samples. Input files that contains disk usage data are generated by using acctdisk(8) or by merging cacct disk output files with other cacct files, using the csaaddc(8) command.

-f Reports full data.

-g Reports group name.

-h Displays headers.

-j Reports number of processes and jobs.

-m Reports CPU breakdown multitasking.

-n Reports prime and nonprime data.

`-w` Reports I/O wait time and I/O wait memory integral.

`-x` Reports blocks transferred and physical and logical I/O.

`-y` Reports SDS data.

`-C` Reports system call and interrupt CPU times.

`-J` Reports job ID.

`-M` Reports Cray MPP system usage statistics. If there is no attached MPP system, the `-M` option reports 0.

NOTES

Zero is a valid value for the number of jobs. If a job is executed with multiple user ID/account ID pairs, the number-of-jobs value is incremented for only one such combination per job.

EXAMPLES

The following example generates a report from a daily accounting file:

```
csacrep -hcw < /usr/adm/acct/sum/data/0203/1315/cacct
```

SEE ALSO

`acctdisk`(8), `csaaddc`(8), `csacon`(8)

UNICOS Resource Administration, Cray Research publication SG–2302

NAME

csadrep – Reports daemon usage

SYNOPSIS

/usr/lib/acct/csadrep [-a] [-A] [-D *level*] [-j] [-n] [-o *ofile*] [-s *sfile*] [-t] [-V *level*]

/usr/lib/acct/csadrep [-A] [-D *level*] [-o *ofile* [-N]] [-s *sfile*]

/usr/lib/acct/csadrep [-a] [-D *level*] [-j] [-n] [-o *ofile*] [-t] [-V *level*] *files*

/usr/lib/acct/csadrep [-D *level*] [-o *ofile* [-N]] *files*

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The csadrep command reports usage of the NQS and tape daemons. Input is either from a session file created by csabuild(8) or from a binary file created by csadrep with the -o option. The files operand specifies the binary files.

The usage report is written to stdout.

The csadrep command accepts three types of options: input, output, and report selection options.

Input Option

The csadrep command accepts the following input option.

-s *sfile* Specifies the name of the session record file. The csabuild(8) command creates this file. The default is /tmp/Super-record.

Output Options

The csadrep command accepts the following output options.

-A Reports all jobs, including those that have not completed. By default, only jobs that have completed are reported.

-D *level* Sets the debugging level. Level 1 is slightly verbose, and level 10 is very verbose. Debug output is written to stderr. By default, debugging is turned off.

-o *ofile* Specifies the name of the binary output file. csadrep can process this file, using the files parameter.

-N Does not generate a usage report. You must use the -o option with this option.

-V *level* Sets the verbose level of the usage report. The levels are 0 through 3. Level 0 is terse, and level 3 is extremely verbose. The default is level 0.

Report Selection Options

The `csadrep` command accepts the following report selection options.

`-a` Reports usage for all daemons. This is equivalent to `-jnt`.

`-j` Reports NQS and interactive job usage. This is the default.

`-n` Reports NQS daemon usage.

`-t` Reports tape daemon usage.

EXAMPLES

Example 1: The following example generates an NQS and tape daemon usage report from the session record file `srec`. The verbose level is set to 3, and binary output file `drep.1` is created.

```
csadrep -a -V 3 -s srec -o drep.1
```

Example 2: The following example generates a terse usage report for all the daemons. Input is from three previously created binary files.

```
csadrep -a drep.1 drep.2 drep.3
```

SEE ALSO

`csabuild`(8), `csanqs`(8)

UNICOS Resource Administration, Cray Research publication SG–2302

NAME

csaedit – Displays, verifies, and deletes records from various accounting files

SYNOPSIS

/usr/lib/acct/csaedit [-a | -x [-t]] [-b *offset*] [-n *nqsfile* | -N *pnqsfile* | -T *tpfile*] [-o *ofile*] [-r *reclist*] [-v]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The csaedit command verifies and outputs accounting records in binary or ASCII form. Only valid records are displayed. You can delete selected records from the output file.

To ensure that all of the bad records were removed, reverify the accounting file after deleting any records. Verification can be done with either the csaedit or csaverify(8) command.

The csaedit command accepts three types of options: input, output, and record selection options.

Input Options

At most, only one input option can be specified. The csaedit command accepts one of the following input options:

-n *nqsfile* Specifies an accounting NQS file that has not been preprocessed. The format of this file is specified in the /usr/include/acct/dacct.h file.

-N *pnqsfile* Specifies a preprocessed NQS accounting file. This file is created by csanqs(8). This is the default with Pnqacct1 as the file name.

-T *tpfile* Specifies a tape accounting file. The format of this file is specified in the /usr/include/acct/dacct.h file.

Output Options

The csaedit command accepts the following output options:

-a Specifies ASCII output. The default is to output binary data.

-o *ofile* Specifies the output file. The default is stdout.

-t Outputs CPU times. If you specify -N, queue wait time is also displayed. You must use this option with either the -a or the -x option.

-v Specifies verbose mode. Verbose output is written to stderr.

-x Specifies no execute mode. Only the records to be deleted are displayed. The selected records are not actually deleted. Output is written to stderr.

Record Selection Options

The `csaedit` command accepts the following record selection options:

`-b` *offset* Specifies the byte offset of the record to be deleted. This offset can be obtained from `csaverify`(8).

`-r` *reclist* Specifies the record numbers of the records to be deleted. *reclist* is a comma-separated list.

EXAMPLES

Example 1: The following example outputs the preprocessed NQS file in ASCII. CPU times are reported.

```
csaedit -N Pnqacct1 -at
```

Example 2: The following example deletes records 2, 10, and 15 from an NQS file. The output is written to file `nqacct.NEW` and verbose output is written to file `err`.

```
csaedit -n nqacct1 -r 2,10,15 -v -o nqacct.NEW 2> err
```

FILES

`/usr/include/sys/accthdr.h`	Defines the accounting header
`/usr/include/acct/dacct.h`	Defines the daemon accounting header

SEE ALSO

`csanqs`(8), `csapacct`(8), `csaverify`(8)

UNICOS Resource Administration, Cray Research publication SG–2302

NAME

csafef2 – Produces summarized session records

SYNOPSIS

/usr/lib/acct/csafef2 [-A] [-s *sessionfile*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The csafef2 command is a template that shows a site how to write a program that summarizes session file records and outputs them as ASCII text. Records are summarized by user name, job ID, and account ID. Information is not reported for records with account IDs of –2 (queued and waiting NQS jobs) and 0. (csafef2 is not intended to execute in its released state. See NOTES.)

The csafef2 command template accepts the following options:

-A Reports both terminated and active sessions. The default is to report ony terminated sessions.

-s *sessionfile* Specifies the session file name. The default is /tmp/Super-record, which is the output from the csabuild(8) command.

NOTES

The csafef2 command is not a stand-alone command, but a template. It executes only after local source modification at sites with source licenses. Due to license restrictions on code included in csafef2, this template available only at source sites.

SEE ALSO

csabuild(8), csafef(8), csaibm(8)

UNICOS Resource Administration, Cray Research publication SG–2302

NAME

csafef – Formats data for front-end accounting system

SYNOPSIS

/usr/lib/acct/csafef [-a] [-s *sessionfile*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The csafef program is a template that shows a site how to build a program that formats data for a front-end accounting system. (It is not intended to execute in its released state. See NOTES.)

The csafef program template accepts the following options:

-a Includes session records only for jobs that have terminated. If you omit -a, records for all jobs are formatted, including those jobs that have not terminated.

-s *sessionfile* Specifies the session file name. The default is /tmp/Super-record, which is the output from the csabuild(8) command.

NOTES

The csafef program is not a stand-alone command, but a template. It executes only after local source modification at sites with source licenses. Due to license restrictions on code included in csafef, these templates are available only at source sites.

BUGS

The -u, -v, and -D, options are allowed also, but they are not used by csafef. These unused options are printed in the csafef usage message.

SEE ALSO

csabuild(8), csafef2(8), csaibm(8)

UNICOS Resource Administration, Cray Research publication SG–2302

NAME

csagcon – Consolidates accounting data for session and pacct files

SYNOPSIS

csagcon [[[-a] [-c] [-j [-N]] [-u]] [-g]] [-d *flags*] [-m *nword2*] [-o *outfile*] [-s *sort_type*] [-A] [-M *nword1*] [-R *request_file*] [-S *srec_file*] [-T *table_file*]

csagcon [[[-a] [-c] [-j [-N]] [-u]] [-g]] [-d *flags*] [-m *nword2*] [-o *outfile*] [-s *sort_type*] [-C] [-M *nword1*] [-R *request_file*] [-S *srec_file*] [-T *table_file*]

csagcon -P [[[-a] [-c] [-j] [-u]] [-g]] [-d *flags*] [-m *nword2*] [-o *outfile*] [-s *sort_type*] [-M *nword1*] [-R *request_file*] [-T *table_file*] *pacct_files*

csagcon -E [-d *flags*] [-m *nword2*] [-o *outfile*] [-M *nword1*] [-O *nrec*] [-R *request_file*] [-T *table_file*] *pacct_files*

csagcon -I [-d *flags*] [-m *nword2*] [-o *outfile*] [-M *nword1*] [-O *nrec*] [-R *request_file*] [-T *table_file*] *pacct_files*

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The csagcon command consolidates the accounting data in either a session file, which is created by the csabuild(8) utility, or in a group of pacct (per-process accounting data) files according to user-specified keys. You may specify the data keys and the fields to be consolidated.

To generate reports from the consolidated data file, use the csagfef(8) command. The csagcon -E and -I options allow you to select data for individual processes. By using either of these options and csagfef, you can produce tailored per-process accounting reports similar to acctcom(1) output.

By default, csagcon consolidates a session file named Session-Record by account ID and user ID. Only data for terminated sessions is collected, and the data items that are consolidated are similar to those produced by csacon(8). The unsorted output is written to a file named gacct.

The five types of csagcon command options are as follows:

- Consolidation options
- Record selection options
- Input options
- Output options
- Performance options

Consolidation Options

Consolidation options cannot be used with the -E and -I options. You can specify multiple options to consolidate the accounting data you have collected, depending upon the capacity of your front-end computer system and the data that is important to report for your site. If you do not specify a consolidation option, the default is to consolidate data for terminated sessions by account ID and user ID (-au options). The csagcon command accepts the following consolidation options:

-a Consolidates records by using account ID as a key.

-c Consolidates records by using job class as a key; job class is either interactive or Network Queuing System (NQS). If this option is used with the -P option, then the job class for all records is interactive.

-g Consolidates records by using group ID as a key. Because not all records have a group ID, you must use this option with at least one of the following consolidation options: -a, -c, -j, or -u.

-j Consolidates records by using job ID as a key.

-u Consolidates records by using user ID as a key.

-A Consolidates all sessions, including those that have not completed. By default, only sessions that have completed are consolidated. This option must be used with the -S option and is mutually exclusive with the -C option.

-C Consolidates only active sessions. By default, only sessions that have completed are consolidated. This option must be used with the -S option and is mutually exclusive with the -A option.

-N Consolidates each portion of an NQS request according to its job ID. This option must be used with the -j option and cannot be used with the -P option. By default, all portions of a request are processed as though they had the same job ID. This option is useful when a request has multiple job IDs, as in the case of rerun requests or requests that use pipeclient or netclient.

Record Selection Options

By default, csagcon consolidates the accounting data. When this occurs, per-process information is no longer available.

The following options generate unconsolidated output. They cannot be used with the -P, -S, or -s consolidation option.

-E Allows access to per-process data found in the pacct eof (end of job) record.

-I Allows access to per-process data found in the pacct base.

Input Options

If you do not specify an input option, the default is to use a session file named Session–Record as the input file (the -S option). The csagcon command accepts the following input options:

`-P` *pacct_file* Lists the `pacct` files to be consolidated. The `-E`, `-I`, `-P`, and `-S` options are mutually exclusive. *pacct_file* is a comma separated list of `pacct` filenames.

`-S` *srec_file* Names the session file to be consolidated. The `-E`, `-I`, `-S`, and `-P` options are mutually exclusive. The default filename is `Session-Record`.

`-T` *table_file* Names the table initialization file. By default, *table_file* is `/usr/lib/acct/table_init`. This option is used only in developing source code. It is recommended that you not use an alternate table initialization file because `csagcon` expects the data variable names defined in this file.

Output Options

If you do not specify output options, the default is to consolidate only the default items and to write the unsorted data to a file named `gacct`. The `csagcon` command accepts the following output options:

`-d` *flags* Specifies the debug flags. The flags are as follows:

Flag	Description
`0000`	No debugging (default)
`0001`	Variable mapping information
`0002`	Prefix mapping information
`0004`	Consolidated data array information
`0010`	Timing information
`0020`	Memory allocation information
`0040`	File information
`0100`	Identifying keys from the input records
`0200`	Output information
`0400`	Tape device group name information

To specify multiple flags, add the numerical values. For example, to produce memory allocation and file information, set the flag to 060 (020 + 040). Debugging output is written to `stderr`.

By default, debugging is disabled.

`-o` *outfile* Names the output file where the consolidated data is written. By default, the output is written to a file named `gacct`.

`-s` *sort_type* Sorts the output file. By default, the output is unsorted. This option cannot be used with the `-E` and `-I` options. Valid *sort_types* are as follows:

sort_type	Action
`acid`	Sorts first by numeric account ID then user ID. Must be used with the `-a` or `-u` option.
`acname`	Sorts first by account name then by username. Must be used with the `-a` or `-u` option.
`jclass`	Sorts first by job class then by job ID. Must be used with the `-c` and `-S` option.

`uid` Sorts first by numeric user ID then by account ID. Must be used with the `-a` or `-u` option.

`username` Sorts first by username then by account name. Must be used with the `-a` or `-u` option.

`-R` *request_file* Names the file that contains a list of data fields to be consolidated. If `-R` is not specified, `csagcon` consolidates the default items, which are listed in *UNICOS Resource Administration*, Cray Research publication SG–2302. These default items are similar to the items that `csacon`(8) consolidates.

Performance Options

You can control how much memory the program allocates each time it reserves a block of memory for various data structures. The `csagcon -d 020` option shows run-time memory allocation information.

When per-process data is generated with the `-E` and `-I` options, you can specify the number of records `csagcon` processes before outputting any data.

The `csagcon` command accepts the following performance options:

`-m` *nword2* Specifies the number of words of memory that the program reserves on all allocations except the first. By default, 131072 words (256 clicks) are allocated.

`-M` *nword1* Specifies the number of words of memory that the program reserves on the first allocation. By default, 393216 words (768 clicks) are allocated.

`-O` *nrec* Specifies the number of per-process data records to process before writing to the output file. By default, 2000 records are processed.

This option can only be used with the `-E` or `-I` options.

EXAMPLES

The following example consolidates the data found in a session file named `Super-record.0815`. Only the default items for terminated sessions are consolidated. The output, which is written to the file `gacct.0815`, is sorted first by user ID, then by account ID.

```
csagcon -S Super-record.0815 -o gacct.0815 -s uid
```

NOTES

Users may require privilege to access the `/dev/kmem` file. If a user does not have the appropriate privilege, `csagcon` will terminate with an error.

FILES

`acct`(5)	Per-process accounting (`pacct`) file.
`/usr/lib/acct/table_init`	Default table initialization file.

SEE ALSO

`acctcom`(1)

`csabuild`(8), `csacon`(8), `csagfef`(8)

UNICOS Resource Administration, Cray Research publication SG–2302

NAME

csagfef – Formats consolidated accounting data

SYNOPSIS

csagfef [-c] [-d *flags*] [-f *infile*] [-v] [*source_file*] ...

csagfef [-d *flags*] [-f *infile*] [-v] [-D *name*[=*def*]] [*source_file*] ...

csagfef -h [-f *infile*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The csagfef generic front-end formatter formats the consolidated accounting data into either an ASCII report or a binary data file. The consolidated data is created by the generic data consolidator, csagcon(8).

The data is formatted based on the specifications found in csagfef source scripts, which are written in a language based on awk. The *source_file* operand specifies the names of these scripts. By default, the source scripts are read from stdin (standard input).

The language is described in detail in *UNICOS Resource Administration*, Cray Research publication SG–2302. It is similar to the language recognized by the tsar(8) command.

The csagfef command accepts the following options:

-c Compiles the source files only. This option is used to debug source files. The input data file is not formatted.

-d *flags* Specifies the debug flags. The flags are as follows:

Flag	Description
0001	Lexical scanning
0002	Expression compilation
0004	Table entry
0010	Code execution
0020	Stack contents
0040	Input file parsing
0100	Symbol table searching
0200	Table allocation

To specify multiple flags, add the numeric values. For example, to enable lexical scanning and code execution debugging, set the flag to 011 (001 + 010).

By default, debugging is disabled.

-f *infile* Specifies the name of the input file to be formatted. The input file was created by csagcon(8). The default input filename is gacct.

`-h` Produces information about the input file including the variable names, constant variables, and number of data records. No source files are compiled. Output is written to `stdout` (standard output).

`-v` Specifies that verbose output be written to the `stderr` file when the source file is processed.

`-D` *name*[=*def*] Defines a symbol name to be used in the source file during execution. *def* may be a number or a character string. Character strings must be delimited by escaped double quotes. (See the EXAMPLES section.)

By default, *def* is defined as the number 1.

source_file ... Specifies source script or scripts to be used to format the input file.

EXAMPLES

In this example, the file `gacct.0815` is being formatted according to the source file `mk_rpt`. The symbol `uname` is defined as the string `user1`.

```
$ csagfef -f gacct.0815 -D uname=\"user1\" mk_rpt
```

FILES

`/usr/src/cmd/acct/src/csa/csagfef/examples`
Directory containing example source scripts

`gacct` The default input file created by `csagcon`(8) and used with the [`-f` *infile*] option

SEE ALSO

`csagcon`(8)

UNICOS Resource Administration, Cray Research publication SG–2302

The AWK Programming Language, by A. V. Aho, B. W. Kernighan, P. J. Weinberger, Addison-Wesley, 1988

NAME

csaibm – Converts session records into IBM format

SYNOPSIS

/usr/lib/acct/csaibm [-a] [-A] [-D *level*] [-o *outfile*] [-r] [-s *sessionfile*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The csaibm command is a template that shows a site how to write a program that converts a session file to IBM format. It is not intended to execute in its released state.

The csaibm command template accepts the following options:

-a Does not translate ASCII strings to EBCDIC. The default is to output all strings in EBCDIC.

-A Converts session records for all jobs, including those that have not terminated. The default is to convert records only for terminated sessions.

-D *level* Sets the debug level. Level 1 is slightly verbose; level 10 is very verbose. By default, debugging is turned off.

-o *outfile* Writes the EBCDIC records to file *outfile*. The default is to write the output to stdout.

-r Does not report rerun portions of an Network Queuing System (NQS) request separately. Data from all portions are added together and written to one ibmiduse record. The default is to write separate usage records for each portion of a rerun NQS request.

-s *sessionfile* Specifies the session file name. The default is /tmp/Super-record, which is the output from csabuild(8).

BUGS

When you specify -r and there are NQS jobs that have rerun portions, ibmnqs records do not have the correct stop time. The first ibmnqs record have the stop time of the entire request. All subsequent ibmnqs records have a stop time of 0. When you do not specify -r, the correct stop times are written.

SEE ALSO

csabuild(8), csafef(8), csafef2(8)

UNICOS Resource Administration, Cray Research publication SG–2302

NAME

csajrep – Prints a job report from the session record file

SYNOPSIS

/usr/lib/acct/csajrep [-b] [-c] [-e] [-h] [-m] [-q] [-t] [-w] [-x] [-y] [-A] [-B] [-C] [-F] [-J] [-L] [-M] [-S *file*] [-T] [-W] [-Z]

/usr/lib/acct/csajrep [-a *acid*] [-b] [-c] [-e] [-h] [-j *jid*] [-m] [-q] [-s *reqid*] [-t] [-u *uid*] [-w] [-x] [-y] [-A] [-B] [-C] [-J] [-L] [-M] [-S *file*] [-T] [-W] [-Z]

/usr/lib/acct/csajrep [-N [-A]] [-S *file*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The csajrep command reports session accounting information from the session file, which is created by the csabuild(8) command.

The csajrep command accepts the following three types of options: input, selection, and printing. If you omit the options, input is read from /tmp/Super-record and all records from each completed session are reported. You can select a subset of sessions by specifying the user ID, account ID, job ID, or Network Queuing System (NQS) request ID for which you seek accounting information.

Input Options

The following option specifies an input file for the job accounting report:

-S *file* Specifies the name of a session file created by csabuild(8). The default file is /tmp/Super-record.

Selection Options

By default, the -a, -j, -s, and -u selection options report entire sessions. The -e option limits the report to records that match the selection criteria. The -e option must be used with at least one of the following options: -a, -j, -s, or -u. You cannot specify the -F option in combination with any of the following options: -a, -j, -s, or -u.

Otherwise, the csajrep command accepts the following selection options:

-a *acid* Specifies a numeric account ID or an account name.

-e Selects only the records that meet the selection criteria as defined by the -a, -j, -s, and -u options. The -e option must be used with at least one of these four options. By default, csajrep outputs all of the records from sessions that contain at least one record that meets the selection criteria.

-j *jid* Specifies a (numeric) job ID.

-s *reqid* Specifies an NQS request number (*reqid*).

`-u` *uid* Specifies a numeric user ID of a user login name.

`-F` By default, `-F` displays all records for completed sessions. This option cannot be used with any of the following options: `-a`, `-e`, `-j`, `-s`, and `-u`. When `-F` is used with the `-A` option, all records for both active and completed sessions are reported. When `-F` is used with the `-Z` option, all records are reported except those with a job ID (*jid*) of 0.

`-N` Selects only NQS requests. Information about each segment of a request is reported by `job_id`/`user_id`/`account_id` combination. For a more complete description of information available on NQS requests, see *UNICOS Resource Administration*, Cray Research publication SG–2302.

`-Z` Ignores records for which the job ID is equal to `0`.

Printing Options

The `csajrep` command accepts the following printing options:

`-b` Reports system billing unit (SBU) usage.

`-c` Reports CPU usage.

`-h` Suppresses report headers.

`-m` Reports multitasking CPU information.

`-q` Reports queue wait time and queue type for NQS jobs.

`-t` Prints summary information.

`-w` Reports the I/O wait time while a process is locked in memory and the memory high-water mark.

`-x` Reports I/O statistics.

`-y` Reports SDS usage statistics.

`-A` Reports both active and completed sessions. By default, only completed sessions are reported.

`-B` Reports process and session starting times.

`-C` Reports system call and interrupt CPU times.

`-J` Reports job ID.

`-L` Puts form feeds at the end of each session.

`-M` Reports Cray MPP system usage statistics. If there is no attached MPP system, the `-M` option reports 0.

`-T` Prints only summary information for each session.

`-W` Reports I/O wait time while a process is not locked in memory.

NOTES

A session may contain multiple user IDs and account IDs, because the user may have executed commands such as su(1) or newacct(1). Also, NQS sessions have multiple job IDs when the request is rerun.

pacct end-of-job records, NQS accounting records, and connect-time records do not contain account IDs. Thus, when the -a and -e options are used together, no end-of-job, NQS, or connect-time information is reported.

Only NQS information is reported when the -s and -e options are used together, because only NQS accounting records contain the NQS request ID.

Accounting records sometimes have a job ID of 0 when Cray system accounting (CSA) cannot determine the correct job ID.

When the -u option is used without the -e option, all records for sessions containing at least one accounting record for the specified user are displayed.

For example, if user1 executes the command rsh cray who from a remote host, then the command /usr/lib/acct/csajrep -u user1 -JBc -S Super-record would produce output similar to:

```
JOB ID   ACCOUNT  LOGIN    COMMAND         START           USER-TIM SYS-TIM
         NAME     NAME     NAME            TIME            [SECS]   [SECS]
======== ======== ======== ======== ==================== ======== ========
     134 Xydev    user1    who      Jul 19 10:10:27 1993    0.008    0.012
     134 System   root     rshd     Jul 19 10:10:27 1993    0.001    0.010
         END OF JOB AT Mon Jul 19 10:10:28 1993
```

The rshd(8) command was executed by root on behalf of user user1; thus it is reported by the csajrep -u option.

When used with -u, the -e option suppresses the printing of all accounting records which are not for the specified user. In the previous example, the command /usr/lib/acct/csajrep -eu user1 -JBc -S Super-record would produce output similar to:

```
JOB ID   ACCOUNT  LOGIN    COMMAND         START           USER-TIM SYS-TIM
         NAME     NAME     NAME            TIME            [SECS]   [SECS]
======== ======== ======== ======== ==================== ======== ========
     134 Xydev    user1    who      Jul 19 10:10:27 1993    0.008    0.012
```

EXAMPLES

Example 1: The following example generates a list of commands by job that user jdoe executed. The list includes job ID, start time, and both terminated and nonterminated jobs in the output:

```
csajrep -u jdoe -ABJ
```

Example 2: The following example prints information about NQS job `4140`:

```
csajrep -s 4140 -BJcqtwx
```

SEE ALSO

`csabuild`(8), `csaline`(8), `csanqs`(8), `csarun`(8), `csatape`(8)

UNICOS Resource Administration, Cray Research publication SG–2302

NAME

csaline – Preprocesses connect-time sessions

SYNOPSIS

/usr/lib/acct/csaline [-l *file*] [-o *file*] [-p] [-t] [-u *file*] [*path*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The csaline command processes a utmp(5) file and outputs a list of connect sessions in ctmp.h format sorted by ending time. This list is written to standard output. You also can obtain line usage, reboot, and boot time information.

The *path* operand specifies the path name of a connect-time accounting file. The default path is /usr/adm/acct/work/Wwtmp. The file names are generated by appending a number to the end of the path name. A file with an appendix of 1 (*path*1) must exist; other files need not exist. For example, the path /w/Wwtmp represents the files /w/Wwtmp0, /w/Wwtmp1, and so on. /w/Wwtmp1 must exist; otherwise, csaline fails.

The csaline command accepts the following options:

-l *file* Writes the line usage summary to *file*. This file contains a summary of line usage showing line name, number of minutes used, percentage of total elapsed time used, number of sessions charged, number of logins, and number of logoffs. This file helps track line usage, identify bad lines, and find software and hardware inconsistencies and errors. Hang-up, termination of login(1), and termination of the login shell each generate logoff records. Thus, the number of logoffs is often three to four times the number of sessions. See the init(8) and utmp(5) man pages for more information.

-o *file* Writes an overall record for the accounting period that gives the starting time, ending time, number of reboots, and number of date changes.

-p Prints the input in ASCII showing the line name, login name, and time. The time is in both number and date/time formats. Processing of the data is not done.

-t Uses the last time found in the input when calculating the connect time for active login sessions. This ensures reasonable and repeatable numbers for noncurrent files. The default is to use the current time.

-u *file* Writes system boot times found in the input file to *file*.

NOTES

If the file `path0` exists, it must be in `ctmp.h` format. All other input files should be in `utmp`(5) format.

EXAMPLES

The following example shows how to extract as much information as possible from file `wtmp`:

```
csaline -t -l lineuse -o reboots /usr/adm/acct/work/1220/1305/Wctime > Pctime1
```

FILES

`/etc/wtmp`	Login records format

SEE ALSO

`init`(8)

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

NAME

csam – Displays system activity data on a dumb terminal

SYNOPSIS

/usr/bin/csam [-f *replayfile*] [-h *host*] [-i *interval*] [-l *logfile*] [-p *passes*] [-d] [-s] [-u] [-C] [-D] [-F] [-H] [-K] [-L] [-M] [-P] [-T] [-W] [-X] [-Y]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The csam command displays various system performance statistics. It uses the curses(3) library to drive the terminal so that many terminal types may be supported. The screen is refreshed every *interval* seconds, for the specified number of *passes*. If *passes* is not specified, csam runs continuously. Specify the system to be monitored with the *host* option. If *host* is not specified, the local system is monitored.

The csam command communicates through rpc(3C) with the sam server, samdaemon(8), running on the *host* system to obtain the information to be displayed. The server can run on a Cray Research system or on a connected operator workstation (OWS).

The csam command checks the size of the screen and the LINES environment variable and can adjust most displays to fill the screen.

The csam command accepts the following options:

-d Sets the debug flag. Debug messages are written to the log file.

-f *replayfile* Specifies a file name to be used for replay of previously recorded sam data. The default is client.recplay. You can create the replay file by using either the csam or xsam(8) command. The data written to the record file are those necessary to produce the displays on the client at the time of recording. An xsam client can record data from several hosts at the same time. Replay is at a 1-second refresh rate, but you can adjust this rate by using the <+> and <-> keys. When it reaches the end of the data in the replay file, csam displays a message at the top of the screen. You can record or replay data by using the record and replay control panel, which you select with the <z> key. The <d> key returns you to normal client displays and the help screen is presented automatically.

-h *host* Specifies the network name of the host to be monitored. If the host is a server running on an OWS, you must specify the network name of the OWS. The default host is the local system.

-i *interval* Sets the refresh rate to *interval* seconds. The default interval is set by the server.

-l *logfile* Specifies the name of the debugging messages log file. The default log file name is client.log.

-p *passes* Specifies the number of times csam will refresh the screen. The default is –1, which gives an infinite display.

`-s` Sets the `user/kernel/idle/wait` option on the kernel display. (This may be the only option for systems with more CPUs than can be displayed on the screen. See the `-K` option.)

`-u` Sets user and system CPU time option for the kernel display. (See the `-K` option.) The `-u` option is the default.

`-C` Selects the configuration display, which shows aspects of the hardware and software configuration of the target system.

`-D` Selects the disk display, which shows each disk device and the transfer rate in the last interval. The transfer rate is presented in Mbyte/s and as a bar graph on a logarithmic scale.

`-F` Selects the logical device cache display. This display is similar to the display provided by the `ldcache`(8) command in refresh mode. To select the next cached file system, press <n>, and then press <r> to reset the display for the current file system.

`-H` Selects the help display, which gives a brief explanation of the various displays and commands available.

`-K` Selects the kernel display. This display has the following parts:

- One part that shows various kernel counters from the `sysinfo`, `syserr`, `syswait`, `pws` tables. Four columns of numbers are displayed; those shown as floating-point numbers are counts per second in the refresh interval, those shown as integers are absolute values. Any number that refers to memory or swap usage is in click units (512 64-bit words).
- One part that displays a number of bar graphs. The first two graphs are the hit rates (read and write) for the system buffer cache. How the remaining graphs are used depends on the screen size, the number of CPUs in the system and the values of the `-s` and `-u` options. If there are enough lines on the screen to display all the CPUs and the `-u` option (the default) is used, the user (`*`) and system (`=`) CPU usage for each of the CPUs is displayed. If the `-s` option is in effect or there are not enough lines on the screen to display all the CPUs, the `kernel/user/wait` and idle percentages for the whole system are displayed.

`-L` Selects the logical device display. This display shows the number of Mbyte/s transferred during the previous interval for each logical device as a number and shows the same information also as a bar graph in a logarithmic scale. File systems cached by the `ldcache`(8) command are displayed on two lines: the first line is marked with the character `C` and shows the data transferred between the cache and the user (display character `*`), the second line is marked with the character `D` and shows the data transferred between the cache and the disk (display character `=`).

`-M` Selects the memory display. A large section to the left shows a map of the common memory usage, low memory in the upper left corner and the high end of memory at the bottom right. Each character on the map represents the number of clicks (512 64-bit words) of memory given by the scale shown in the legend box. Different characters indicate the state of the process (or one of the processes) occupying that space. A capital letter indicates the start of a process and a small letter indicates a continuation of the same process. For example, the string `SSSRrrr` indicates three (or more) small sleeping processes followed by one larger process that is runnable.

To the right of the common memory map is a legend box and a box showing percentage memory and swap utilization. The `oversub` field is the oversubscription factor for central memory (in other words, the amount of memory that would be required to hold all the processes in the system).

At the bottom of the display are a series of counters showing the number of processes in various states, both in memory and swapped.

`-P` Selects the process display. This display is similar to the output from the `ps`(1) command. This display is not refreshed. You can scroll to the next screen by using the <n> key and you can reset to the first full screen by using the <p> key. You can also specify a numeric process ID followed by a <RETURN>, and the display will switch to a refreshing snap of the process selected. If `csam` does not find the process or the process terminates while the snap display is active, then `csam` displays an appropriate message and the screen returns to an updated process display.

`-T` Selects the tape display, which shows each tape device and the transfer rate in the last interval. The data is presented as Mbyte/s and as a bar graph on a logarithmic scale. (Deferred implementation.)

`-W` Selects the swap map display. This display is identical to the memory map display, except that the map section maps the swap device instead of central memory. On all Cray Research systems with partitioned swap devices the start of each swap partition (after the first) is indicated by the character `*`.

`-X` Selects the top processes display. This display shows a sorted list of processes, each line containing a process name, its process ID, the percentage of one CPU used by that process in the last interval and a bar graph. The bar graph represents the type of CPU usage. User CPU time is marked with an `*`, idle time with a `.`, and system CPU time with an `=`. A total of all active processes, including those that do not fit on the screen, but excluding idle processes, is shown at the top of the screen. Multitasked processes are marked with the character `M`.

`-Y` Selects the system call display. This display shows a sorted list of system call activity, including the total time spent processing system calls in the last interval, shown as a percentage of one CPU, and the total number of calls per second. Each line on the display provides information for one system call and contains the number of calls per second and the percentage of time spent processing those calls. This second percentage is a percentage of the total system time figure. Therefore, if the total system time is 25% and the system time for the `write` system call is 20%, then the time spent in the `write` system call is 5% of one CPU.

Interactive Input

After `csam` is running on your terminal, you can use the following keys to change displays, move within a display, increase or decrease refresh rates, and exit:

Key	Description
<c> or <C>	Selects host system configuration display.
<d> or <D>	Selects disk display.
<e> or <E>	Quits program.
<f> or <F>	Selects ldcache display.
<g> or <G>	Leaves single-step mode.
<h> or <H>	Selects help screen.
<k> or <K>	Selects kernel display.
<l> or <L>	Selects logical device display.
<m> or <M>	Selects memory display.
<n> or <N>	Advances to next page (disk, ldcache, logical device, process, and tape displays).
<p> or <P>	Selects process display.
<q> or <Q>	Quits program.
<r> or <R>	Resets bar graphs and ldcache display.
<s> or <S>	Selects `user/kernel/wait/idle` usage on kernel display.
<t> or <T>	Selects tape display. (Deferred implementation.)
<u> or <U>	Selects user and system CPU usage on kernel display.
<w> or <W>	Selects swap map display.
<x> or <X>	Selects top process display.
<y> or <Y>	Selects system call display.
<z> or <Z>	Selects record/replay control panel.
<+>	Increases refresh interval by 1 second.

<-	Decreases refresh interval by 1 second. The refresh interval cannot be lowered below the refresh rates set by the server.
<.>	Enters single-step mode. This freezes the display until the space bar is pressed to advance the screen.
Space bar	Advances display in single-step mode.
Numeric input	Selects a process ID for the snap display. This is accepted only in the process display. The ID must be terminated by a carriage return. The erase and kill characters are accepted during this numeric input.

BUGS

Because system tables can change while they are being read, `csam` occasionally may produce a misleading display. This usually is corrected during the next refresh.

SEE ALSO

`ldcache`(8), `sam`(8), `samdaemon`(8), `xsam`(8)

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

UNICOS Resource Administration, Cray Research publication SG–2302

NAME

csanqs – Preprocesses the NQS accounting files

SYNOPSIS

/usr/lib/acct/csanqs [-n *file*] [-t] [-D *level*] [*pathname*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The csanqs command processes the Network Queuing System (NQS) accounting files and generates one output record for each segment of an NQS job.

The csanqs command accepts the following options and operand:

-n *file* Writes a list of null (rejected) NQS jobs to the specified *file*.

-t Calculates queue wait time as the difference between the current time and the time the request entered the pipe queue for requests that are queued and have never executed. The wait time is recalculated when the process begins execution. If you use -t with recycled NQS data, the queue wait time for queued but never executed requests will be erroneous. The wait times will be corrected when the requests start execution and csanqs is run again. For requests that have executed on a CPU and are currently queued, or when -t is omitted, queue wait time is reported only after the request begins execution on a CPU.

-D *level* Sets the debug level. Level 1 is slightly verbose; level 10 is very verbose. By default, debugging is turned off.

pathname Specifies the path name of the NQS accounting file. The default path name is /usr/adm/acct/work/Wnqacct. To generate the file names, append a number to the path name. A file with an appendix of 1 (*pathname*1) must exist. Other files need not exist. For example, the path name /w/nqacct represents the files /w/nqacct0, /w/nqacct1, and so on. /w/nqacct1 must exist; otherwise, csanqs fails.

NOTES

The csanqs command calculates queue wait time by subtracting the time the request entered the pipe queue from the time the request began executing on your Cray Research system. For a request that has been checkpointed, the amount of time the request spent checkpointed is also considered queue wait time. The queue wait time includes the amount of time that queues were stopped or disabled, the time that the Cray Research system was down, and the time the request spent waiting because it was submitted with the qsub -a option (see qsub(1)).

EXAMPLES

A typical usage of `csanqs` is as follows:

```
csanqs /usr/adm/acct/work/1201/1305/Wnqacct > Pnqacct1
```

FILES

`/usr/adm/acct/day/nqacct*`	Current NQS accounting files

SEE ALSO

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

UNICOS Resource Administration, Cray Research publication SG–2302

NAME

csapacct – Verifies and deletes records from a pacct file

SYNOPSIS

/usr/lib/acct/csapacct [-o *offset*] [-r *recnum*] [-v] *inputfile outputfile*

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The csapacct command reads records from a pacct file and verifies each record. Records that are not valid are ignored. Only "good" records, (those not containing truncated or otherwise corrupted data) are written to the output file.

If you specify a byte offset, the record that resides at this location is deleted. Byte offsets of "bad" records (those containing data that is not valid) are obtained from the csaverify(8) program.

To be sure that all of the bad records were deleted, reverify the pacct file after deleting records. Verification can be done with either the csapacct or the csaverify(8) command.

The csapacct command accepts the following options and operands:

-o *offset* Specifies the byte offset of the record to be deleted. The offset can be obtained from csaverify(8).

-r *recnum* Specifies the record number of the base pacct record to be deleted.

-v Specifies verbose mode. Verbose output is written to stderr.

inputfile Input file. This file must be in acct(5) format.

outputfile Output file.

EXAMPLES

The following example shows how to verify and delete bad records from pacct file pacct1. The output is written to file pacct.NEW. Verbose mode is turned on.

```
csapacct -v pacct1 pacct.NEW
```

SEE ALSO

csaedit(8), csaverify(8)

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

NAME

csaperiod – Runs periodic accounting

SYNOPSIS

/usr/lib/acct/csaperiod [-e *MMDDhhmm*] [-r] [-s *MMDDhhmm*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The csaperiod command directs the processing of the daily consolidated accounting files, which are created by the csarun(8) command. The csaaddc(8) command merges the daily files into one file. The csacrep(8) command then generates a report based on the merged file.

The progress of csaperiod is recorded in the pdactive file. When an error is detected, a message is written to the operator, and mail is sent to root and adm. Further data processing is halted.

The cron(8) command usually initiates csaperiod.

The csaperiod command accepts the following options:

-e *MMDDhhmm*
Selects consolidated accounting data generated at or before the specified date, *MMDDhhmm*.

-r Removes the daily data files after processing is done. The default is to leave the daily data files in the /usr/adm/acct/sum/data directory.

-s *MMDDhhmm*
Selects consolidated accounting data generated at or after the specified date, *MMDDhhmm*.

NOTES

By default, csaperiod processes all the /usr/adm/acct/sum/data/* files. The mail recipients (root and adm) can be changed by modifying the MAIL_LIST parameter in the /etc/config/acct_config file.

FILES

/etc/config/acct_config	Accounting configuration file
/usr/adm/acct/fiscal/data/*MMDD*/*hhmm*/cms	Periodic command usage data in cms record format
/usr/adm/acct/fiscal/data/*MMDD*/*hhmm*/pdacct	Periodic condensed data files
/usr/adm/acct/fiscal/rpt/*MMDD*/*hhmm*/rprt	Periodic report files
/usr/adm/acct/nite/pdactive	Log file

`/usr/adm/acct/nite/E*`*MMDDhhmm*	Error messages
`/usr/adm/acct/sum/data/`*MMDD*`/`*hhmm*`/cacct`	Daily condensed data files

SEE ALSO

`acctcms`(8), `cron`(8), `csaaddc`(8), `csacon`(8), `csacrep`(8), `csarun`(8)

UNICOS Resource Administration, Cray Research publication SG–2302

NAME

csaperm – Changes group ID and permissions of accounting files

SYNOPSIS

/usr/lib/acct/csaperm [-v]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The csaperm command sets the group IDs to adm (administrator) in the /etc/csainfo and /etc/wtmp files, and in all the accounting files in the /usr/adm/acct directory. It also sets the file permissions so that a user in the group adm with permission bit acct set can run accounting. This makes it unnecessary to have super-user permissions to run accounting.

The csaperm command accepts the following option:

-v Specifies verbose mode. File names are reported as changes are made.

FILES

/etc/csainfo	System boot times
/etc/wtmp	Login information
/usr/adm/acct/*	Accounting directories and files

SEE ALSO

udbgen(8)

UNICOS Resource Administration, Cray Research publication SG–2302

NAME

csarecy – Recycles unfinished sessions into next accounting run

SYNOPSIS

/usr/lib/acct/csarecy [-r] [-s *file*] [-u *path*] [-A] [-C *path*] [-D *level*] [-N *path*] [-P *path*] [-T *path*]

/usr/lib/acct/csarecy [-r] [-s *file*] [-u *path*] [-C *path*] [-D *level*] [-N *path*] [-P *path*] [-R] [-T *path*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The csarecy command retrieves job information from the accounting files of the current accounting period and places it in the accounting files of the next accounting period. The default input is /tmp/Super-record. csabuild(8) marks unfinished accounting jobs (those that do not terminate in a given period of system activity). csarecy takes these records from the session file and puts them into the next period's accounting files directory. This process is repeated until the job finishes.

csarecy also prints reports about unfinished accounting jobs, in the following format:

```
SYSTEM BOOT TIME STARTING AT MMDDYY hh.mm
PRESERVED ACCOUNTING SESSIONS (jobs that are continued)
=====================================================
  JOB ID     USERS       ACCOUNTS              STARTED
---------    -----     -------------  ------------------------
```

The csarecy command accepts the following options:

-r Produces a report on all recycled jobs.

-s *file* Specifies session record *file* (from csabuild(8)) as the input file. The default is /tmp/Super-record.

-A Asks you whether you want to select each job for recycling. When this option is used, you should run csarecy interactively. The -R option cannot be used with this option.

-C *path* Specifies the path name of the output file for connection accounting information. The system adds a 0 to the end of the file name, so the actual file name is *file*0. The default is /usr/adm/acct/work/Pctime.

-D *level* Sets debugging level. Level 1 is slightly verbose; level 10 is very verbose.

-N *path* Specifies the path name of the output file for NQS accounting information. The system adds a 0 to the end of the file name, so the actual file name is *file*0. The default is /usr/adm/acct/work/Pnqacct.

`-P` *path* Specifies the path name of the output file for `pacct` accounting information. The system adds a 0 to the end of the file name, so the actual file name is *file*`0`. The default is `/usr/adm/acct/work/Wpacct`.

`-R` Produces report only; does not recycle jobs. The `-A` option cannot be used with this option.

`-T` *path* Specifies the path name of the output file for tape subsystem accounting information. The system adds a 0 to the end of the file name, so the actual file name is *file*`0`. The default is `/usr/adm/acct/work/Ptpacct`.

`-U` *path* Specifies the path name of the output file for `uptime` accounting information. The system adds a 0 to the end of the file name, so the actual file name is *file*`0`. The default is `/usr/adm/acct/work/Puptime`.

NOTES

By default, recycled jobs are ignored by most accounting programs.

SEE ALSO

`csaaddc`(8), `csabuild`(8), `csacon`(8), `csacrep`(8), `csafef`(8), `csajrep`(8), `csaline`(8), `csanqs`(8), `csaperiod`(8), `csarun`(8), `csatape`(8)

UNICOS Resource Administration, Cray Research publication SG–2302

NAME

csarun – Processes the daily accounting files and generates reports

SYNOPSIS

/usr/lib/acct/csarun [-A] [-V *level*] [*MMDD* [*hhmm* [*state*]]]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The csarun command, usually initiated by cron(8), directs the processing of the daily accounting files. csarun processes connect, daemon, and process accounting files.

If errors occur, csarun does not damage the active accounting files. It records its progress by writing descriptive diagnostic messages to the active file. When an error is detected, a message is written to the operator, and mail is sent to root and adm. Further data processing is halted.

Before invoking csarun on a new accounting period, ensure that the previous invocation of csarun has completed successfully. If this is not done, information about unfinished sessions will not be accurate.

A series of lock files are used to protect against reinvocation. The lock and lock1 files prevent simultaneous invocation.

The csarun command accepts the following options and operands:

-A Accounts for both terminated and active jobs. By default, only terminated jobs are reported. csarun does not recycle active sessions.

-V *level* Controls verification level of accounting data files.

MMDD [*hhmm* [*state*]]
Sets month, day, hour, minute, and state for which csarun will rerun the accounting. If csarun is restarted, the month and day are necessary; other portions are optional.

The csarun command breaks its processing into separate, restartable states using statefile to remember the last state completed. It accomplishes this by writing the state name into statefile. csarun looks in statefile to determine what must be processed next. The states are executed in the following order:

State Name	Description
SETUP	Moves active accounting files into a work directory.
WTMPFIX	Verifies the integrity of the /etc/wtmp file (see utmp(5)). If necessary, date changes are corrected.
VERIFY	Verifies the integrity of the data files.

PREPROC	Preprocesses the /etc/wtmp, Network Queuing System (NQS) accounting, and tape accounting files.
ARCHIVE1	User exit that executes a site-dependent accounting program or script to archive the raw and preprocessed accounting files.
BUILD	Organizes the accounting data into a session record file.
ARCHIVE2	User exit that executes a site-dependent accounting program or script to archive the session record file.
CMS	Generates command summaries.
REPORT	Generates daily accounting reports.
DREP	Generates daemon usage report.
FEF	User exit that executes a site-dependent accounting program or script to format the session record file into a format that is suitable for use on a front end.
USEREXIT	User exit that executes a site-dependent accounting program or script.
CLEANUP	Cleans up temporary files and exits.

Before restarting csarun after a failure, check the active file for diagnostics, then fix any corrupted data files such as pacct or wtmp. The lock files must be removed before csarun can be restarted. If csarun is restarted, you must specify the *MMDD* operand, which specifies the month and day for which csarun will rerun the accounting. The entry point for processing is based on the contents of statefile. To override this entry point, include the desired state on the command line to designate where processing should begin.

NOTES

The mail recipients (root and adm) can be changed by modifying the MAIL_LIST parameter in the /etc/config/acct_config file. You also can change the other parameters defined in the accounting configuration file for your site.

csarun checks the number of free blocks in the file system that contains the accounting files to ensure that it consists of more than 500 blocks. By default, csarun assumes that the file system is /usr; if this is not the case, change symbol ACCT_FS in /etc/config/acct_config accordingly. To change the minimum number of free blocks on ACCT_FS (the default is 500), modify MIN_BLKS in the configuration file.

To remove bad records in accounting data files encountered by csarun, use csaedit(8), csaverify(8), and csapacct(8).

BUGS

If possible, do not restart `csarun` in the SETUP state. Instead, run SETUP manually and restart `csarun` by using the following command line:

`csarun` *MMDD hhmm* WTMPFIX

If `csarun` terminates abnormally and leaves the lock files in place, the next execution of `csarun` will remove these locks, but it also will terminate abnormally.

EXAMPLES

Example 1: The following example shows how to start `csarun`:

```
nohup csarun 2> /usr/adm/acct/nite/fd2log &
```

Example 2: The following example shows how to restart `csarun` at the state specified in `statefile`:

```
nohup csarun 0601 1345 2>> /usr/adm/acct/nite/fd2log &
```

Example 3: The following example shows how to restart `csarun` at a specific state:

```
nohup csarun 0601 1345 BUILD 2>> /usr/adm/acct/nite/fd2log &
```

FILES

`/etc/config/acct_config`	Accounting configuration file
`/etc/wtmp`	Connect time information
`/usr/adm/acct/day/*`	Directory that contains current accounting files
`/usr/adm/acct/nite/active`	Record of accounting progress
`/usr/adm/acct/nite/clastdate`	Record of last date and time that accounting ran
`/usr/adm/acct/nite/lock`	Lock file that prevents simultaneous invocation
`/usr/adm/acct/nite/lock1`	Lock file that prevents simultaneous invocation
`/usr/adm/acct/nite/statefile`	Record of last state that `csarun` was working on or completed
`/usr/adm/acct/sum/data/*`	Directory that contains daily condensed data files
`/usr/adm/acct/sum/rpt/*`	Directory that contains daily accounting reports
`/usr/adm/acct/work/*`	Directory that contains temporary files from daily accounting
`/usr/lib/acct/csa.archive1`	Site-generated user exit program or script to be executed during the `ARCHIVE1` state.
`/usr/lib/acct/csa.archive2`	Site-generated user exit program or script to be execute during the `ARCHIVE2` state.

`/usr/lib/acct/csa.fef`	Site-generated user exit program or script to be execute during the `FEF` state.
`/usr/lib/acct/csa.user`	Site-generated user exit program or script to be execute during the `USEREXIT` state.

SEE ALSO

`acctcms`(8), `cron`(8), `csaaddc`(8), `csabuild`(8), `csacon`(8), `csacrep`(8), `csadrep`(8), `csaedit`(8), `csafef`(8), `csajrep`(8), `csaline`(8), `csanqs`(8), `csapacct`(8), `csaperiod`(8), `csaperm`(8), `csarecy`(8), `csaverify`(8), `fwtmp`(8)

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

`acct`(5), `utmp`(5), `wtmp`(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

csasocket – Processes socket accounting data

SYNOPSIS

/usr/lib/acct/csasocket [-d *family*] [-f *address*:*port*] [-g *address*] [-G *group*] [-h] [-i *interface*] [-l *address*:*port*] [-n] [-N *program name*] [-o *output file*] [-p *protocol*] [-P *pid*] [-r *start*:*end*] [-s] [-S] [-t *type*] [-u *user*] [-v] [-1] [*input file*]

IMPLEMENTATION

All UNICOS systems

DESCRIPTION

The csasocket command processes all socket accounting records contained in the given *input file*, which defaults to /usr/adm/acct/day/soacct.

The csasocket command accepts the following options and operands:

-d *family*
: Identifies the domain (i.e. address family) of the sockets to be processed. Any record found for a socket which does not match *family* will be ignored.

  family can be specified as either the numerical value of the desired domain, or as one of the following:

Family	Description
inet	Process any/all Internet sockets.
unix	Process any/all UNIX domain sockets.

-f *address*:*port*
: Identifies the foreign address and/or port of the sockets to be processed. Any record found for a socket which was not connected to *address* and/or *port* will be ignored (this includes any socket which was never connected, see NOTES below).

  If *port* is specified, the delimiter ":" must be included.

  If the -d option was used to specify a specific domain other than inet, this option is not allowed.

  address can be specified as either the foreign host's name or its internet address in dot notation.

  port can be specified as either the service's name assigned to the port in /etc/services or the numerical value of the port.

`-g` *address*
: Identifies the gateway used by the sockets to be processed. Any record found for a socket which did not use *address* as a gateway will be ignored (this includes those sockets for which a gateway was not known, see NOTES below). Since the route a socket uses may change, this option may not show all sockets that actually used *address* as a gateway.

  If the `-d` option was used to specify a specific domain other than `inet`, this option is not allowed.

  address can be specified as either the gateway's host name or its internet address in dot notation.

`-G` *group*
: Identifies the group ID of the sockets to be processed. Any record found for a socket whose group ID is not *group* will be ignored.

  group can be specified as either the group's name, as found in `/etc/group`, or its numerical value.

`-h`
: Indicates that the contents of each accounting record's header, found in *input file*, are to be written to standard error.

`-i` *interface*
: Identifies the interface used by the sockets to be processed. Any record found for a socket which did not use *interface* to access the network will be ignored (this includes those sockets for which an interface was not known, see NOTES below). Since the interface a socket uses may change, this option may not show all sockets that actually used *interface* to access the network.

  If the `-d` option was used to specify a specific domain other than `inet`, this option is not allowed.

  interface must be specified as the name of the interface followed by its unit number. That is, it must be specified as it is displayed by `netstat`(1B).

`-l` *address*:*port*
: Identifies the local address and/or port of the sockets to be processed. Any record found for a socket which was not opened at *address* and/or *port* will be ignored (this includes those sockets for which the local address/port were not known, see NOTES below).

  If *port* is specified, the delimiter ":" must be included.

  If the `-d` option was used to specify a specific domain other than `inet`, this option is not allowed.

  address can be specified as either the local host's name or its internet address in dot notation.

  port can be specified as either the service's name assigned to the port in /etc/services or the numerical value of the port.

`-n` Indicates that any value displayed should not be translated into that value's equivalent name. This option applies to the following values that may be displayed:

Value	Value displayed if `-n` specified
address	The internet address in dot notation.
port	The numerical value of the port.
group	The numerical value of the group.
user	The numerical value of the user's ID.

`-N` *program name* Identifies the program's name that last issued a close on the sockets to be processed. Any record found for a socket which was not closed by *program name* will be ignored.

Only the first 16 bytes of the program's name will be used to match which sockets are to be processed.

`-o` *output file* Identifies the name of the file where all of the records that were processed are to be written. Any record found in *input file* that matched all of the selection criterion specified on the command line will be written (appended) to *output file*.

If *output file* does not exist, it will be created with the owner and group of the executing user and permissions 0666 masked with the executing user's `umask`(1). `csaperm`(8) can be used to change this file's group ID and/or permissions.

`-p` *protocol* Identifies the protocol (e.g. TCP) of the sockets to be processed. Any record found for a socket which was opened using a protocol other than *protocol* will be ignored.

protocol can be specified as either the numerical value of the desired protocol or as one of the following:

Protocol	Description
`icmp`	Process any/all ICMP protocol sockets.
`tcp`	Process any/all TCP protocol sockets.
`udp`	Process any/all UDP protocol sockets.

`-P` *pid* Identifies the process ID of the last process executing which closed the sockets to be processed. Any record found for a socket whose process ID is not *pid* will be ignored.

`-r` *start*:*end* Identifies the date/time range during which the sockets to be processed were opened and closed. Any socket which was opened before *start* or was closed after *end* will be ignored.

If *end* is specified, the delimiter ":" must be included.

Both *start* and *end* can be specified as follows:

[*cc* [*yymmddH*]]*HMM*

Where each character's meaning is consistent with the `date`(1) command. Any string of digits from 3 to 10, or 12, may be specified. The digit's location in the string is interpreted as indicated above.

`-s` Indicates that a summary of all sockets processed is to be written to standard output following all other information written.

`-S` Indicates that only the summary of all sockets processed is to be written to standard output. That is, information detailing each record processed will not be displayed.

`-t` *type* Identifies the type (e.g. stream) of the sockets to be processed. Any record found for a socket type other than *type* will be ignored.

type can be specified as either the numerical value of the desired type or as one of the following:

Type	**Description**
raw	Process any/all raw sockets.
dgram	Process any/all datagram sockets.
stream	Process any/all stream sockets.

`-u` *user* Identifies the owner of the sockets to be processed. Any record found for a socket whose owner is not *user* will be ignored.

user can be specified as either the user's name, as found in the UDB, or the numerical value of the user's ID.

`-v` Indicates that the following information is to be written to standard error:

1. A detailed description of the criterion used to select sockets for processing.
2. Whether any data (and the amount) was found in the *input file* that was ignored. This would indicate that some data existed in the *input file* between the two valid accounting records indicated.
3. Whether any accounting record (and the amount) was truncated. This would indicate that the start of another accounting record was found imbedded in the indicated record.
4. Whether any accounting record, which was truncated, was repaired by padding the record with binary zeros or skipped.
5. Whether any accounting error record was found in the *input file*. The information contained in this record will be displayed.

If this option is not specified, any data found to exist between any two valid accounting records is silently ignored, any truncated record is silently repaired (i.e. it is reconstructed to it original length by appending bytes of binary zeros or it is skipped), and any error record found is silently ignored.

`-1` Indicates that that the information pertaining to each record written to standard output is to be written as one line per record in *keyword=value*, whitespace delimited, pairs.

The following keywords will be contained in each record:

Keyword	Description
`FAMILY`	The socket's address family/domain.
`TYPE`	The socket's type.
`PROTOCOL`	The socket's protocol.
`CREATED`	The date/time (*ccyymmddHHMMSS.xxx*) the socket was created.
`DESTROYED`	The date/time (*ccyymmddHHMMSS.xxx*) the socket was destroyed.
`OPTIONS`	The socket's options in hexidecimal.
`UID`	The socket's owner's UID.
`GID`	The socket's group ID.
`PROGRAM`	The program executing when the socket was destroyed.
`PID`	The process ID of the executing process that destroyed the socket.
`R_PEEKED`	The number of bytes of data returned to the executing program when the `MSG_PEEK` flag was set.
`R_COUNT`	The number of reads/receives performed on the socket.
`R_BYTES`	The number of bytes of data received on the socket.
`W_COUNT`	The number of writes/sends performed on the socket.
`W_BYTES`	The number of bytes of data sent on the socket.

The following, additional, keywords will be contained in each socket that existed in the internet domain:

Keyword	Description
`L_ADDRESS`	The socket's local address.
`L_PORT`	The socket's local port number.
`F_ADDRESS`	The socket's foreign address.
`F_PORT`	The socket's foreign port number.
`GATEWAY`	The gateway that the socket used.
`INTERFACE`	The interface that the socket used.

input file Identifies the file to be read in and scanned for socket accounting records matching the selection criterion specified on the command line.

The default file is `/usr/adm/acct/day/soacct`.

The `csasocket` command reads each accounting record found in the specified *input file* and, for each "socket" accounting record found, determines whether it matches ALL of the selection criterion specified on the command line. If ANY field does not match the given criterion, the record is ignored. For example:

```
csasocket -d inet -t stream
```

Will result in all stream sockets that existed in the internet domain being processed. While:

```
csasocket -d inet
```

Will result in all sockets that existed in the internet domain being processed.

Only those records selected for processing will be written to the *output file*, provided the `-o` option is specified on the command line.

The contents of each record selected will be displayed on standard output, provided the `-S` option is not specified.

If either the `-s` or `-S` option is specified, a summary of all records selected will be written to standard output. This summary will include the minimum, maximum, average and totals of all numerical values.

NOTES

Only the accounting records that matched all of the selection criterion specified on the command line are written (appended) to the *output file*, whether or not the `-v` option is specified. Error records will not be written to the *output file*.

You can use the `csaperm`(8) command to change all of the accounting files' group ID and permissions as necessary.

All data, contained in each of the accounting records written, reflects the state of that socket at the point in time the last program having the socket open closes it. As a result, some of the fields may not contain any information because that information was not known at the time the accounting record was written. For example, in general a UDP protocol socket is never connected (see `connect`(2)) which will result in its foreign address/port being zero. This would also be true for a TCP protocol socket which is listening for incoming connections (see `listen`(2)). In general, the following fields may not be defined at the point in time the accounting record is written:

1. The foreign address/port (`-f`) will be zero for any socket which was never connected, or for which the connection was broken (for example, the executing program received a `SIGPIPE` signal; see `signal`(2)).
2. The gateway (`-g`) used will be zero for any socket which does not have a foreign address identitied, or for which the route to that foreign address was lost.
3. The interface (`-i`) used will be zero for any socket which does not have a foreign address identified, or for which the interface was taken down while the socket was open.

4. The local address/port (`-l`) will be zero for any socket which was never bound (see `bind`(2)), or for which the connection was broken (for example, the executing program received a `SIGPIPE` signal).

   This occurs because the state of the socket can change over time. As noted earlier, the information contained in each accounting record describes the state of the socket at the time the last `close`(2) is issued. As a result, the information contained in an accounting record could show that the socket did send/received some data, yet not have any/all of these fields defined. For example, a UDP socket which is used to send/receive packets from several different addresses would not have a foreign address/port, gateway, nor interface identified. A socket which was once connected but is terminated abnormally may not even have the local address/port identified, as this information could be cleared out before the program issues the `close`(2).

RETURN VALUES

This program will exit with a zero exit status if no unrecoverable error occurred. If any unrecoverable error was encountered, this program will display an error message and terminate with a non-zero exit status.

EXAMPLES

Example 1:

The following command extracts all internet socket accounting records found in `/usr/adm/acct/day/soacct` and writes them to `soinet`. Also, information describing any invalid data, and any error records, found is displayed on standard error. Only a summary of the records written to `soinet` is displayed on standard output.

```
csasocket -o soinet -v -d inet -S
```

Example 2:

The following command displays the contents of all socket accounting records found in `soinet` along with a summary of all the records found.

```
csasocket -s soinet
```

FILES

```
/etc/group
/etc/hosts
/etc/services
/etc/udb.public
/usr/adm/acct/day/soacct
```

SEE ALSO

`csa`(8), `csaperm`(8)

`date`(1), `umask`(1), `netstat`(1B) in the *UNICOS User Commands Reference Manual*, Cray Research publication SR–2011

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

`gid2nam`(3C), `nam2gid`(3C), `nam2uid`(3C), `uid2nam`(3C) in the *UNICOS System Libraries Reference Manual*, Cray Research publication SR–2080

NAME

csaswitch – Checks the status of, enables, and disables process, daemon, and record accounting

SYNOPSIS

Positional formats:

/usr/lib/acct/csaswitch [-D *level*] [[-o *options*] | [-t *threshold*]] on *name pathname*
/usr/lib/acct/csaswitch [-D *level*] off *name*
/usr/lib/acct/csaswitch [-D *level*] check *name*
/usr/lib/acct/csaswitch [-D *level*] [-a] status

Non-positional formats:

/usr/lib/acct/csaswitch [-D *level*] [[-o *options*] | [-t *threshold*]] -c on -n *name* -p *pathname*
/usr/lib/acct/csaswitch [-D *level*] -c off -n *name*
/usr/lib/acct/csaswitch [-D *level*] -c check -n *name*
/usr/lib/acct/csaswitch [-D *level*] [-a] -c status

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The csaswitch command checks the current status of, enables, and disables process (kernel), daemon, and record accounting.

The csaswitch command accepts the following options or operands:

-D *level* Sets the debug level for displaying messages during the command's execution. These messages are written to stderr (standard error).

The level can be 0 to 10 with a higher number increasing the number of messages displayed. The default is 0, which results in no messages being displayed.

[-c] on Enables the indicated accounting method.

[-c] off Disables the indicated accounting method.

[-c] check Displays the current status of the indicated accounting method.

[-c] status Displays the current status of all accounting methods.

[-n] *name* Specifies the accounting method being checked, enabled, or disabled.

Valid daemon names are:

Name	Description
dm	Data migration accounting
kernel	System (process) accounting

`nqs`	Network Queuing System (NQS) accounting
`socket`	Socket (network) accounting
`tape`	Tape accounting

Valid record names are:

Name	**Description**
`dio`	Device I/O accounting
`mpp`	Device MPP accounting
`mt`	Multi-tasking accounting
`perf`	Performance accounting
`sds`	Secondary data storage accounting

[`-p`] *pathname* Specifies the file name the accounting method is to use to hold accounting data.

`csaswitch` creates this file if it does not exist already, sets the owner and group of the file to `adm`, and sets the mode to 0660.

`-t` *threshold* (Implementation deferred) Specifies the number of CPU seconds or the memory size below which accounting records will not be written.

`-o` *options* Specifies additional options to be processed by the accounting method being enabled. If more than one option is specified, the options must be whitespace delimited and the entire set enclosed in quotes.

This option is currently only supported for socket accounting.

For socket accounting, the following options are valid:

Option	Field	Description
`inet`	Family	Account for all Internet sockets.
`unix`	Family	Account for all UNIX domain sockets.
`raw`	Type	Account for all raw sockets.
`dgram`	Type	Account for all datagram sockets.
`stream`	Type	Account for all stream sockets.
`icmp`	Protocol	Account for all ICMP protocol sockets.
`tcp`	Protocol	Account for all TCP protocol sockets.
`udp`	Protocol	Account for all UDP protocol sockets.

An accounting record will be written for each socket closed that matches this selection criterion. If no options are specified, an accounting record will be written for every socket as it is closed.

Only one option of a given field type can be specified, and the options can be combined to

reduce the sockets selected for accounting (i.e., specifying `"inet stream"` will produce an accounting record for each Internet stream socket closed, while just specifying `stream` will produce an accounting record for every stream socket closed no matter which address family the socket was created in).

`-a` Specifies the status of all accounting methods is to be displayed, not just a predefined subset.

NOTES

The positional and nonpositional formats of the command can be mixed such that the positional parameters appear last and in the order shown above, if the associated option (i.e. `-c`, `-n` or `-p`) is not specified on the command line.

The check and status functions do not require any special privileges.

You must be a privileged user, and authorized for group `adm`, in order to enable/disable accounting. The privileges required are documented in `acctctl`(2).

The commands `turnacct`(8) and `turndacct`(8) can be used to front end this command in order to enable and/or disable accounting.

You can use the `csaperm`(8) command to change all of the accounting files' group ID and permissions as necessary.

RETURN VALUES

For a `check` request, `csaswitch` returns 0 if the accounting method is currently `ON`. If the accounting method is currently `OFF` or its status could not be determined, a non-zero value is returned.

For all other requests, `csaswitch` returns 0 if the request was successfully performed; otherwise, a non-zero value is returned.

EXAMPLES

Example 1: The following command turns on kernel (process) accounting such that all data will be written to file `/usr/adm/acct/day/pacct`:

```
csaswitch on kernel /usr/adm/acct/day/pacct
```

Example 2: The following command checks the status of kernel (process) accounting:

```
csaswitch -c check -n kernel
```

Information similar to the following is displayed:

```
#       Accounting status for Wed  Wed Nov 16 13:36:47 1994
#       Name    State   Value
        kernel  On
```

Example 3: The following command turns on socket (network) accounting such that all data associated with sockets opened for communication over the Internet will be written to file `/usr/adm/acct/day/sockacct`:

```
csaswitch -o inet on socket /usr/adm/acct/day/sockacct
```

Example 4: The following command checks the status of socket (network) accounting:

```
csaswitch check socket
```

Information similar to the following is displayed:

```
#       Accounting status for Wed  Wed Nov 16 13:36:47 1994
#       Name    State   Value
        socket  On      inet
```

Example 5: The following command checks the status of all accounting:

```
csaswitch -a -c status
```

Information similar to the following is displayed:

```
#       Accounting status for Wed Nov 16 13:38:46 1994
#       Name    State   Value
        kernel  On
        nqs     On
        tape    On
        dm      Off
        socket  On      inet
        cray1   Off
        cray2   Off
        site1   Off
        site2   Off

        dio     On
        mpp     On
        mt      On
        perf    On
        sds     On
        mem     Off
        time    Off
        cray3   Off
        cray4   Off
        site3   Off
        site4   Off
```

SEE ALSO

`csaperm`(8), `turnacct`(8), `turndacct`(8), `udbgen`(8)

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