NAME

rlogin – Invokes the remote login

SYNOPSIS

/usr/ucb/rlogin *rhost* [-d] [-e*c*] [-l *username*] [-8] [-E] [-L] [-S *tos*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The rlogin utility connects your terminal on the current local host system to the remote host system *rhost*. Your remote terminal type is the same as your local terminal type (as specified in your TERM environment variable). All echoing occurs at the remote site; therefore, the rlogin is transparent (except for delays). Flow control through <CONTROL-s> and <CONTROL-q> and flushing interrupt input and output are handled properly. A line of the form ~. disconnects from the remote host; ~ is the escape character. A line of the form ~<CONTROL-z> suspends the rlogin process, and a line of the form ~<CONTROL-y> suspends the send portion of the rlogin process, but allows output from the remote system. A line of the form ~z is the same as ~<CONTROL-z>.

The rlogin utility accepts the following options:

rhost	Specifies the remote host.
-d	Uses setsockopt(2) to turn on socket debugging on the TCP sockets that are used for communication with the remote host.
-e*c*	Specifies an escape character (*c*). No space can separate this option flag (-e) and the new escape character (*c*).
-l *username*	Specifies the account name (*username*) to use when logging in to the remote machine. Default is the current account name.
-8	Allows the transmission of 8-bit data.
-E	Stops any character from being recognized as an escape character. When used with the -8 option, this provides a completely transparent connection.
-L	Allows the rlogin process to be run in -opost mode (see stty(1)).
-S *tos*	Sets the IP Type-of-Service (TOS) option for the connection to the value *tos*, which can be a numeric TOS value or a symbolic TOS name found in the /etc/iptos file.

NOTES

The system configuration can require the `/etc/hosts.equiv` and `.rhosts` files each to contain a match for the originating host, and also require the remote user and local user names to match.

The `rlogin` requests are validated to ensure that the remote host or workstation security levels and compartments, as defined in the network access list (NAL), are within the security level range and are authorized compartments for the UNICOS system. The user security values are set to the most restrictive boundary conditions defined by the NAL and the user database (UDB).

The `rlogin` process also validates the user's right to access the UNICOS system from the host. Access to the UNICOS system is granted or denied based on the workstation access list (WAL) check the login process performs.

The results of user validation are recorded in the security log.

BUGS

Not enough terminal characteristics are propagated.

FILES

`$HOME/.rhosts`	Remote machine
`/etc/hosts`	TCP/IP host name database

SEE ALSO

`remsh`(1B), `stty`(1), `telnet`(1B)

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

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

TCP/IP Network User's Guide, Cray Research publication SG–2009

Software Overview for Users, Cray Research publication SG–2052

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

NAME

rls – Releases reserved tape resources

SYNOPSIS

```
rls -a [-f] [-n] [-s]
rls -d dvn|all [-f] [-k] [-n] [-s]
rls -p pathname [-f] [-k] [-n] [-s]
```

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The rls utility releases resources reserved with the rsv(1) utility and breaks the association between the device and the *pathname* (see the -p and -P options of the tpmnt(1) utility).

You must specify -a, -d, or -p.

The rls utility accepts the following options:

-a — Releases all resources. -a is not valid with -d, -k, or -p. See the CAUTIONS section.

-f — Specifies that any scratch volumes from the autoloader scratch pool used during this tape session will be returned to the scratch pool for reuse.

-n — Specifies no-unload status for the tape. This option has the same effect as the -u option on the tpmnt(1) utility. This option is useful when a tape is used repeatedly.

-s — Specifies that any scratch volumes from the autoloader scratch pool used during this tape session will not be returned to the scratch pool. The user will keep these volumes.

-d *dvn*|all — Releases specified devices. For *dvn*, specify all to release all the used devices or specify a list of device names separated by a colon (:) so that only devices identified are released (see the tpstat(1) utility for information on how to find the device name). You cannot use the -d option with the -p option.

-k — Keeps resource privilege. If you specify the -k option, the process limit and current reservation are not decremented. You must specify a path name or a device to be released. You may issue an additional tpmnt(1) command without issuing an rsv(1) command.

-p *pathname* — Specifies a path name previously identified in a tpmnt(1) utility that has a -p or -P option. The request causes the tape equipment, which is associated with the path, along with the privilege to use the equipment, to be released back to the resource pool. You cannot use the -p option with the -d option.

CAUTIONS

The `rls -a` utility line releases all tapes reserved by your process ID and reserved tape resources, including reserved tape resources used by the processes running in the background. The `rls` utility releases resources that are closed; all other resources are put into a `releaseending` state until a `close`(2) system call has been executed for those resources. To release reserved tape resources for a specific process, use the `-p` or `-d` option.

EXIT STATUS

If `rls` completes successfully, 0 is returned; otherwise, a nonzero value is returned. Where possible, this exit status code is normalized to the last three digits. Exit status values are documented in the *Tape Subsystem User's Guide*, Cray Research publication SG–2051.

EXAMPLES

The following examples illustrate different uses of the `rls` utility:

Example 1: The `rls` utility releases devices `tape03` and `tape01`:

```
rls -d tape03:tape01
```

Example 2: The `rls` utility releases all of the devices that have been used:

```
rls -d all
```

Example 3: The `rls` utility releases the tape device that has the path name `tapfile`, which was specified with the `-p` option on `tpmnt`(1):

```
rls -p tapfile
```

Example 4: The `rls` utility releases the tape device that has the path name `tapfile`, allowing you to keep the tape resource and to perform another `tpmnt`(1) following it:

```
rls -p tapfile -k
```

Example 5: The `rls` utility releases all tape resources, does not unload any mounted volumes, and specifies that the autoloader scratch volumes will be saved for the user:

```
rls -a -n -s
```

SEE ALSO

`rsv`(1), `tpmnt`(1), `tprst`(1), `tpstat`(1)

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

Tape Subsystem User's Guide, Cray Research publication SG–2051

NAME

rm, rmdir – Removes files or directories

SYNOPSIS

```
rm [-f] [-i] [-r] [-R] files
rmdir [-p] [-s] dirnames
```

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4
AT&T extension (`rmdir -s` option)

DESCRIPTION

The `rm` utility removes the entries for one or more files from a directory. If an entry was the last link to the file, the file is destroyed. To remove a file, you must have write permission in its directory, but do not need read or write permission on the file itself. (See the note about MAC-protected files in the **NOTES** section.)

If a file has no write permission and the standard input is a terminal, the name and the full set of permissions (in octal) for the file are printed, followed by a question mark. This is a prompt for confirmation. If the answer begins with a lowercase `y` (for yes), the file is deleted; otherwise, the file remains.

If the standard input is not a terminal, `rm` will operate as if the `-f` option were in effect.

The `rmdir` utility removes the specified directories, which must be empty.

The `rm` utility accepts the following options:

`-f` Removes all specified files (ignores write-protection) in a directory without prompting the user. In a write-protected directory, or when MAC prevents writing the file, files are not removed, regardless of their permissions, and an error message is issued.

If a designated file is a directory, and the `-r` and `-R` options were not specified, an error message is issued, and `rm` will continue with any remaining file operands.

Any previous occurrences of the `-i` option are ignored.

`-i` Confirms removal of all files occurs interactively. To remove a file, your response must begin with a lowercase `y`. This option overrides the `-f` option and remains in effect even if the standard input is not a terminal.

Any previous occurrences of the `-f` option are ignored.

`-r`
`-R` Recursively removes any directories and subdirectories in the argument list. The directory is emptied of files and removed. Usually, the user is prompted for removal of any write-protected files in the directory. The write-protected files are removed without prompting if the `-f` option is used, or if the standard input is not a terminal and the `-i` option is not used. (See the note about MAC protected files in the NOTES section.)

files Files to be removed.

If you try to remove a nonempty, write-protected directory, `rm` will fail (even if you use the `-f` option), causing an error message.

If a directory to be removed is the current directory, `rm` silently ignores the current directory name and does not remove it.

The `rmdir` utility accepts the following options:

`-p` Lets you remove directory path *dirname* if it is empty, and its parent directories if they become empty. A message is printed on standard output as to whether the whole path is removed or part of the path remains for some reason.

`-s` Suppresses the message printed on standard error when `-p` is in effect.

dirnames Specifies the names of the directories to be removed.

NOTES

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

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

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

When multi-level security is activated, mandatory access control (MAC) rules can prevent removing a file if MAC would prevent writing the file or the directory that contains it.

EXIT STATUS

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

0 If the `-f` option was not specified, all the named directory entries were removed successfully. If the `-f` option was specified, all the existing named directory entries were removed successfully.

>0 An error occurred.

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

0 Each directory entry specified was removed successful.

>0 An error occurred.

SEE ALSO

`mkdir`(1)

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

NAME

rmdel – Removes a delta from an SCCS file

SYNOPSIS

rmdel -r *SID files*

IMPLEMENTATION

All Cray Research systems

STANDARDS

XPG4

DESCRIPTION

The rmdel utility removes the delta specified by the source identifier (SID) from each named Source Code Control System (SCCS) file. The delta to be removed must be the newest (most recent) delta in its branch in the delta chain of each named SCCS file. In addition, the SID-specified delta must not be that of a version being edited for the purpose of making a delta (that is, if a *p-file* (see get(1) exists for the named SCCS file, the SID-specified delta must *not* appear in any entry of the *p-file*).

The rmdel utility accepts the following option and operands:

-r *SID* Specifies the SID (SCCS IDentification) level of the delta to be removed.

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

If you make a delta, you can remove it; if you own the file and directory, you can remove a delta.

EXIT STATUS

The rmdel utility exits with one of the following exit values:

0 Successful completion.

>0 An error occurred.

MESSAGES

Use help(1) for explanations.

EXAMPLES

The following example removes delta `1.2` from the file `s.example.c`:

```
rmdel -r1.2 s.example.c
```

FILES

`x.`*file*	See `delta`(1)
`z.`*file*	See `delta`(1)

SEE ALSO

`admin`(1), `cdc`(1), `comb`(1), `delta`(1), `get`(1), `help`(1), `prs`(1), `sact`(1), `sccsdiff`(1), `unget`(1), `val`(1), `vc`(1), `what`(1)

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

NAME

rmgr – Provides an interface to the Unified Resource Manager (URM) daemon

SYNOPSIS

rmgr [-c *directive*] [-D] [-h *hostname*] [-I *directory*] [-l *username*] [-s *socket*]

rmgr [-D] [-h *hostname*] [-I *directory*] [-l *username*] [-s *socket*] [*file*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The rmgr utility accepts the following options:

-c *directive* Invokes any rmgr *directive*. When the -c option is used, the *file* argument is not allowed and the rmgr utility terminates after processing the -c *directive*.

-D Debug mode; used for testing purposes only. Writes internal information to the stderr file.

-h *hostname* Names the host on which the target URM is running. The default host is the local host. An environment variable, RMGR_HOSTNAME, may name the target host.

-I *directory* Specifies an alternate directory in which to look for include files. If the file named on an include directive is not found using its name as written, rmgr checks for the file in the *directory* named with -I. Only one -I option can be used.

-l *username* Specifies a different user name by which to connect to URM. Using this option will result in an anonymous connection to URM (nonprivileged). For the connection to be accepted, the specified *username* must be contained under the URM /admin/anonymous/ node.

-s *socket* Specifies a socket (or service found in /etc/services) name or port number on which the target URM is listening. The default service name is urm. An environment variable, RMGR_SOCKET, may name the socket or port number.

file Names a command file. If a name is not given, rmgr will read from the stdin file. If stdin is connected to an interactive device, a prompt will be issued so the user will know that input is needed. If the -c option is used, you cannot specify a *file*.

The rmgr interface includes the following subcommands, some of which may be used only by an authorized administrator. Subcommands can be specified with either an initial uppercase letter or an initial lowercase letter (for example, both Quit and quit are valid). Only the lowercase form is shown here.
NOTE: Additional subcommands exist but are undocumented; these undocumented subcommands are reserved for use by Cray Research.

`delete` *object* (Authorized administrator only) Removes objects from the object tree. To delete an object, you must have write permission to that object. The directive to delete object `/val/myval` is as follows:

```
delete /val/myval
```

The following restrictions apply to what can be deleted:

The root node cannot be deleted.
No object with an existing synonym can be deleted.
No object with a lock can be deleted.
No resource object assigned to any job can be deleted.
A node with child objects cannot be deleted.
Only objects you own, or to which you have write permission, can be deleted.

`include` *file* Switches input file to *file*. After the named file has been read to the end-of-file, reading reverts to the line following this line. Includes can be nested up to 30 levels deep. To include a file named `setup/times`, use the following directive (the file name must appear in quotes):

```
include "setup/times"
```

`quit` Disconnects `rmgr` from the URM daemon and terminates the `rmgr` session.

`repeat` As a prefix to a directive, `repeat` causes `rmgr` to reissue a URM directive every 10 seconds and display the result. For example, to see the machine load changing:

```
repeat view /machine/load
```

`set` *parameter* Changes URM configuration parameter. Parameters include the following:

Parameter	Description
`basenode`	Changes the start of a name search from the root of the object tree to some other starting node (similar to the UNICOS `cd`(1) command). When this directive is set, any object name not beginning with a slash character (`/`) will be found starting at the named node, rather than the root node. An example of this directive follows: `set basenode` *object_name* The named object must be of type `Node` and must exist in the object tree. If the named object is removed while it is the base node, the base node is changed to root.

`defperm` (Authorized administrator only) Sets the default permission for newly created objects. When a user is connected to URM, the default permission is set to `rwr-`. This directive allows that default to be altered. Permissions are read from left to right as "owner read, owner write, other read, other write." All four permission fields must always be stated. Examples of this directive include the following :

```
set defperm rwrw
set defperm r---
```

The first directive sets the default permission for new objects to read and write for every user. This means that anyone can change or delete this object. The second directive makes new objects have owner read only permission. The owner is allowed to remove an object regardless of its permission and the URM administrator can do anything with any object, regardless of its permission.

`lock` (Authorized administrator only) Sets the delete lock on an object. A locked object may not be removed and there is no provision to remove a lock. The lock is intended for essential objects having to do with machine loading information which, for performance reasons, have pointer references from various internal places in URM. An example of this directive follows:

```
set lock /machine/target/memory
```

If any child object of a node is locked, no node higher in the object tree (closer to the root) can be removed.

`log` (Authorized administrator only) Enables logging of directives and informative and error messages. Logging is enabled by default. A log message is always written when this directive occurs:

```
set log
```

`log` *directory* (Authorized administrator only) Enables logging and switches the log file to reside in *directory*. If the path names are different, the current log is terminated with a message and closed, and a new log file is opened in *directory*. Log file names cannot be changed. Error messages appear if *directory* is the same as the current path or if the new path is inaccessible or full. If a log file of today's name already exists in *directory*, new messages will be appended to it.

`nolog` (Authorized administrator only) Disables logging. A log message is written only when this directive changes the state from logging to no logging:

```
set nolog
```

`perm` (Authorized administrator only) Changes the permission of an existing object. Only the URM administrator or the owner of an object can change the permission of that object. For example, for an existing object named `/val/xyz`, its permission can be changed using the following directive:

```
set perm rw-- /val/xyz
```

No matter what its prior permission, the object now has `rw--` permission.

`stopdaemon` (Authorized administrator only) Sends a directive to URM to terminate, then `rmgr` executes `quit`.

`view` [*name*|*type*] *option*

Displays options. The optional *name* or *type* argument can be used to display the name or type, respectively, of an object. The following options are valid:

Option	Description
`eshare` *nnn* [*lnode_ID*]	Shows effective share for user name *nnn*
`help`	Shows a list of all `view` options. NOTE: The `view help` command displays additional `view` options that are undocumented; these undocumented options are reserved for use by Cray Research.
`jlist`	Shows the Joblist table
`jobs` *nnn*	Shows jobs for user name *nnn*
`jobs` *uuu*	Shows jobs for user ID (UID) *uuu*
`jpath`	Shows the Jobpath table
`jselect`	Shows the job selector
`restart` *nnn*	Shows `chkpnt` images for user name *nnn*
`restart` *nnn n*	Shows detailed content of `chkpnt` image number *n*
`restart` *uuu*	Shows `chkpnt` images for user ID *uuu*
`rpath`	Shows the Respath table
`sds`	Shows SDS management values
`tpath`	Shows the Timepath table
`users`	Shows the users chain
`/`*xxx*	Shows object or node chain `/`*xxx* (for example, `view /hosts`)

*/xxx/**	Shows objects below */xxx* (for example, `view /urm`)
xxx	Shows node relative to the current node

object = *exp* Value-type objects may be given a value when they are declared, but can also have their value changed, using the value assignment directive. Value-type objects include numeric values (`Float` or `Int`) and string values (`Str`). Time-type objects may not be given a value in this way.

If an integer object named `/val/myval` existed, to set its value to 1234, use the following directive:

```
/val/myval = 1234
```

If the object is type `Str`, only quoted strings and other objects of type `Str` can appear in the expression. You can assign a value to the object `/val/mystring` as follows:

```
/val/mystring = "A string"
```

The quotation marks will not appear in the strings.

To change the minimum rank for a batch job, use the `usetjob`(8) command.

Checkpointing

URM can be configured to checkpoint individual sessions if requested by the owner of the session. The session owner can use the `chkptint`(1) utility to request either checkpointing at shutdown (interactive sessions only) or periodic checkpointing (either batch or interactive sessions).

Checkpointing is disabled by default. To allow users to checkpoint their sessions, the system administrator must set the URM configuration parameters that control checkpointing type (shutdown or periodic), frequency, interval type (CPU or clock), and length of time the checkpoint file is retained. See *UNICOS Resource Administration*, Cray Research publication SG–2302, for information on enabling checkpointing with `rmgr`.

Checkpointing is done on an individual session basis only. The `chkptint`(1) utility has one required option of the form `-s` *sec*, which specifies the requested checkpoint frequency in seconds. When URM is running in checkpoint-only-at-shutdown mode, all that is required is that `chkptint -s` be specified as a nonzero number. However, to enable periodic checkpointing, you must specify the interval in seconds as the *sec* argument to the `-s` option.

To turn off the automatic or periodic checkpoint request for that session, specify `chkptint -s 0`. The `chkptint`(1) utility can be made part of a script or placed in your `.cshrc` or `.profile` file.

The `view restart` subcommands apply to saved interactive sessions only. NQS jobs that use the `chkptint`(1) command are checkpointed and restarted by NQS, even though URM requested that the job be checkpointed at the appropriate time.

The `view restart` *nnn* subcommand works as follows:

```
rmgr-> view restart kcz
Restart images belonging to User <kcz>, UID 343 on path /ptmp/urm/chkpnt/kcz
        <1> 03201253.1520
        <2> 03211303.1520
        <3> 03231552.616
        <4> 03251613.616
        <5> 03281602.616
        <6> 03301623.616
User <kcz> has 6 restart images.
```

In the `view restart` *nnn n* subcommand, the *n* is the number in angle brackets (<>). This number is also used to specify which `chkpnt` file to restart and which `chkpnt` file to delete.

The form of the `chkpnt` file name is *MMddhhmm.nnnn*, that is, month(*MM*), day(*dd*), hour(*hh*), minute (*mm*), and the session ID of the job (*nnnn*).

To determine whether or not a restart file is a candidate for restart, use the `view restart` *nnn n* subcommand, which will display the processes and how much time has been used.

To dispose of `chkpnt` files you no longer need, use the `delete restart` *nnn n* subcommand. A `view restart` *nnn* or `view restart` *uuu* subcommand must be executed immediately before invoking the `delete` subcommand, as follows:

```
rmgr-> delete restart kcz 1
Deleted restart image <1> for user kcz
rmgr->
```

To restart a saved interactive session, first view the available `chkpnt` files by using the `view restart` *nnn* or `view restart` *uuu* subcommand, then, while in the same `rmgr` invocation, use the `restart` *nnn n* subcommand. If the restart is successful, the current session will be replaced by whatever was in the `chkpnt` file.

When a session is restarted, it resumes execution where it left off. For interactive sessions, this can sometimes be at an unexpected point, and the resulting output can be confusing. For example, if checkpointing occurred in the middle of the output of a `man`(1) command, the restarted session resumes output exactly where it left off. If an interactive session was checkpointed while waiting for input to a prompt, no prompt is displayed when the session is restarted.

See *UNICOS Resource Administration*, Cray Research publication SG–2302, for more information on checkpointing and restarting sessions.

NOTES

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

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

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

EXAMPLES

To review the current value for each URM object:

```
#rmgr
rmgr-> view /urm
LNr-r-  0    0 Sep  8 15:39 <machine       > -> node
LNrwr-  0    0 Sep  8 15:39 <Debug         > -> node
LVrwr-  0    0 Sep  8 15:39 <prevrun_boost > Float, 6.000000
LVrwr-  0    0 Sep  8 15:39 <entitle_wt    > Float, 0.500000
LVrwr-  0    0 Sep  8 15:39 <usage_wt      > Float, 0.500000
LVrwr-  0    0 Sep  8 15:39 <tape_wt       > Float, 0.500000
LVrwr-  0    0 Sep  8 15:39 <share_wt      > Float, 0.500000
LVrwr-  0    0 Sep  8 15:39 <service_wt    > Float, 0.500000
LVrwr-  0    0 Sep  8 15:39 <sds_wt        > Float, 0.500000
LVrwr-  0    0 Sep  8 15:39 <petime_wt     > Float, 0.500000
LVrwr-  0    0 Sep  8 15:39 <pe_wt         > Float, 0.500000
LVrwr-  0    0 Sep  8 15:39 <mem_wt        > Float, 0.500000
LVrwr-  0    0 Sep  8 15:39 <cpu_wt        > Float, 0.500000
LVrwr-  0    0 Sep  8 15:39 <bb_wt         > Float, 0.500000
LVrwr-  0    0 Sep  8 15:39 <age_wt        > Float, 0.500000
LVrwr-  0    0 Sep  8 15:39 <restart_switch>  "Force"
LVrwr-  0    0 Sep  8 15:39 <chkpnt_switch >  "No"
LVrwr-  0    0 Sep  8 15:39 <min_interval  >   Int, 1800
LVrwr-  0    0 Sep  8 15:39 <interval_type >  "Clock"
LVrwr-  0    0 Sep  8 15:39 <retain_chkpnt >   Int, 432000
LVr-r-  0    0 Sep  8 15:39 <shutdown      >   Int, 0
LVr-r-  0    0 Sep  8 15:39 <shut_done     >   Int, 0
LVrwr-  0    0 Sep  8 15:39 <restart_cmd   >  "UPATH/irstart"
LVrwr-  0    0 Sep  8 15:39 <chkpnt_path   >  "PPATH/chkpnt"
LVrwr-  0    0 Sep  8 15:39 <chkpnt_cmd    >  "UPATH/intchkpt"
LVrwr-  0    0 Sep  8 15:39 <sds_suspend   >  "UPATH/sdsspnd"
LVrwr-  0    0 Sep  8 15:39 <share_to_go   >   Int, 896
LVrwr-  0    0 Sep  8 15:39 <share_eval    >   Int, 900
LVrwr-  0    0 Sep  8 15:39 <timeout_user  >   Int, 600
LVrwr-  0    0 Sep  8 15:39 <sleep_time    >   Int, 10
LVrwr-  0    0 Sep  8 15:39 <sds_residence >   Int, 900
LVrwr-  0    0 Sep  8 15:39 <sched_delay   >   Int, 10
LVrwr-  0    0 Sep  8 15:39 <init_wait     >   Int, 1800
LVrwr-  0    0 Sep  8 15:39 <info_delay    >   Int, 10
```

```
LVrwr-  0    0 Sep  8 15:39 <share_policy  >  "Standard"
```

From within `rmgr`, to change the value of one of these URM objects (`sds_residence`, for example):

```
rmgr-> /urm/sds_residence = 100
```

SEE ALSO

`cd`(1), `chkptint`(1), `ustat`(1)

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

UNICOS Resource Administration, Cray Research publication SG–2302

NAME

rpcgen – Generates code to implement Remote Procedure Call (RPC) protocol

SYNOPSIS

rpcgen *infile*
rpcgen [-c] [-h] [-l] [-m] [-o *outfile*] [*infile*]
rpcgen [-s *transport*] [-o *outfile*] [*infile*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The rpcgen compiler is a tool that generates C code to implement an RPC protocol. The input to rpcgen is a language with striking similarity to C, known as Remote Procedure Call Language (RPCL).

Typically, rpcgen is used as shown in the first synopsis, in which it takes an input file and generates four output files. If *infile* is named proto.x, rpcgen generates a header file in proto.h, xdr routines in proto_xdr.c, client-side stubs in proto_svc.c, and server-side stubs in proto_clnt.c.

When you do not want to generate all of the output files, but only a particular one, use the other synopses.

The input can contain C-style comments and preprocessor directives. Comments are ignored; the directives are simply stuffed uninterpreted into the output header file.

To customize eXternal Data Representation (XDR) routines, leave data types undefined. For every data type that is undefined, rpcgen assumes that a routine exists with xdr_ prepended to the name of the undefined type.

The rpcgen compiler accepts the following options:

infile	Specifies the input file.
-c	Compiles XDR routines.
-h	Compiles C data definitions (a header file).
-l	Compiles into client-side stubs.
-m	Compiles into server-side stubs, but does not generate a main routine. This option is useful for doing callback routines and for writing your own main routine to do initialization.
-o *outfile*	Specifies the name of the output file. If you omit *outfile*, standard output is used (-c, -h, and -s modes only).
-s *transport*	Compiles a server into server-side stubs by using the given transport. The supported transports are udp and tcp. You can invoke this option more than once to compile a server that serves multiple transports.

The following summary of RPCL syntax, which is used for `rpcgen` input, is to aid comprehension, rather than an exact statement of the language.

Primitive Data Types

RPCL primitive data types are as follows:

```
[unsigned] char
[unsigned] short
[unsigned] int
[unsigned] long
unsigned
float
double
void
bool
opaque
string
```

Except for the added `opaque`, `bool`, and `string` types, RPCL primitive data types are identical to those of C. The `rpcgen` compiler converts `bool` declarations to `int` declarations in the output header file (literally it is converted to `bool_t`, which has been defined through `#define` to be an `int`). `opaque` data types are converted to arrays of `chars`. You can declare `opaque` data as either a fixed- or variable-length array. C has no intrinsic `string` type, and it uses a null-terminated `char *` by convention. The RPCL keyword `string` is compiled into a `char *` in the output header file. Strings can be declared as having either a maximum length (for example, `string msg<80>;`) or an arbitrary length (for example, `string msg<>;`). Also, `void` declarations can appear only inside `union` and `program` definitions. Rather than typing the prefix `unsigned`, you can use abbreviations `u_char`, `u_short`, `u_int`, and `u_long`.

Declarations

RPCL allows only four kinds of declarations. Following is the format of each of these declarations:

Simple declaration: *type-name object-ident*

Pointer declaration: *type-name* `*`*object-ident*

Vector declaration: *type-name object-ident* `[`*size*`]`
The maximum *size* is specified between the angle brackets; you can omit *size*, indicating that the array can be of any size; *size* can be either an integer or a symbolic constant.

Variable-length-array declaration:

type-name variable-ident `<`*value*`>`
type-name variable-ident `< >`

Because variable-length arrays have no explicit syntax in C, these declarations are compiled into structures. For example, the declaration:

```
int heights <12>;
```

gets compiled into the following structure:

```
struct {

        u_int heights_len;

        int *heights_val;

} heights;
```

The number of items in the array is stored in the `_len` component, and the pointer to the array is stored in the `_val` component.

Type Definitions

The only way to generate an XDR routine is to define a type. For each type *zetype* you define, a corresponding XDR routine named *xdr_zetype* exists.

To define a type, you can use any of the following:

```
typedef
enumeration-def
structure-def
discriminated-union-def
program-def
const-def
```

The `typedef`, `enumeration-def`, and `structure-def` definitions are very similar to their C namesakes. C does not have a formal type mechanism to define variable-length arrays, and XDR unions are quite different from their C counterparts. Program definitions (`program-def`) are not type definitions in the same sense as the others, but they are useful nonetheless. The following bulleted paragraphs describe each of the type definitions and provide a syntax:

- `typedef`
  With RPCL, you cannot declare multidimensional arrays or pointers to pointers inline, unless you use `typedef`. The syntax for an XDR `typedef` is as follows:

  typedef:

  `typedef` *declaration* `;`

The *object-ident* part of *declaration* is the name of the new type; the *type-name* part is the name of the type from which it is derived.

- `enumeration-def`
  The syntax is as follows:

  `enum` *enum-ident* `{`
  enum-list
  `};`

  enum-list:
  enum-symbol-ident [= *assignment*]
  enum-symbol-ident [= *assignment*] , *enum-list*

The *assignment* variable can be either an integer or a symbolic constant. If there is no explicit assignment, the implicit assignment is the value of the previous enumeration plus 1. If not explicitly assigned, the first enumeration receives the value of 0.

- `structure-def`
  The syntax is as follows:

  `struct` *struct-ident* `{`
  declaration-list
  `};`

  declaration-list:
  declaration;
  declaration ; *declaration-list*

You cannot nest XDR definitions. For example, the following is an `rpcgen` error:

```
struct dontdoit {
        struct  ididit {
                int oops;
        } sorry;
        enum ididitagain { OOPS, WHOOPS } iapologize;
};
```

- `discriminated-union-def`
  The syntax is as follows:

union *union-ident* switch (*discriminant-declaration*) {
case-list
[default : *declaration* ;]

};

case-list:
case *case-ident* : *declaration* ;
case *case-ident* : *declaration* ; *case-list*

discriminant-declaration:
declaration

The union definition looks like a cross between a C-union and a C-switch. Following is an example:

```
union net_object switch (net_kind kind) {
case MACHINE:
        struct sockaddr_in sin;
case USER:
        int uid;
default:
        string whatisit;
};
```

The preceding example compiles into the following structure:

```
struct net_object {
        net_kind kind;
        union {
                struct sockaddr_in sin;
                int uid;
                char *whatisit;
        } net_object;
};
typedef struct net_object net_object;
```

The name of the union component of the output structure is the same as the name of the type itself.

- program-def

The syntax is as follows:

```
program program-ident {
        version-list
} = program-number ;

version-list:
        version
        version version-list

version:
        version version-ident {
                procedure-list
        } = version-number ;

procedure-list:
        procedure-declaration
        procedure-declaration procedure-list

procedure-declaration:
        type-name procedure-ident (type-name) = procedure-number ;
```

The following example shows a program definition. To create a server that can get or set the date, you can use the following declaration:

```
program DATE_PROG {
        version DATE_VERS {
                date DATE_GET(timezone) = 1;
                void DATE_SET(date) = 2;        /* Greenwich mean time */
        } = 1;
} = 100;
```

In the header file, this compiles into the following:

```
#define DATE_PROG 100
#define DATE_VERS 1
#define DATE_GET 1
#define DATE_SET 2
```

The client program should use these `define` statements to reference the remote procedures.

- `const-def`
  RPCL constants are symbolic constants that can be used wherever an integer constant is used. The syntax is as follows:

  `const` *const-ident* `=` *integer.*

An RPCL symbolic constant is used in the following array size specification:

```
const DOZEN = 12;
```

In the header file, this compiles into the following:

```
#define DOZEN 12
```

When using `rpcgen` to compile your server, the server interfaces to your local procedures by expecting a C function with the same name as that in the program definition, but it is in all lowercase letters and followed by the version number. The following is a local procedure that implements `DATE_GET`:

```
date *      /* always returns a pointer to the results */
date_get_1(tz)
    timezone *tz;       /* always takes a a pointer to the arguments */
{
        static date d;  /* must be static! */
        /*
         * figure out the date
         * and store it in d
         */
        return(&d);
}
```

XDR recursively frees the argument after getting the results from your local procedure; therefore, you should copy from the argument any data that you will need between calls. However, XDR neither allocates nor frees your results. You must handle their storage.

Inference Rules

You can set up suffix transformation rules in `make`(1) for compiling XDR routines, client and server-side stubs, and header files. The convention is that RPCL protocol files have the extension `.x`. An example of `make` rules to do this is as follows:

```
.SUFFIXES: .x
.x.c:
        rpcgen -c $< -o $@
.x.h:
        rpcgen -h $< -o $@
.x.l:
.x.m:
                .
                .
                .
```

BUGS

Name clashes can occur when you are using program definitions, because the apparent scoping does not really apply. You can avoid most of these by giving unique names for programs, versions, procedures, and types.

Nesting is not supported. As a workaround, you can declare at top-level, and use their name inside other structures to achieve the same effect.

SEE ALSO

Remote Procedure Call (RPC) Reference Manual, Cray Research publication SR–2089

NAME

rsv – Reserves tape resources

SYNOPSIS

rsv [-m *message_file*] [-t] [*resources*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The rsv utility reserves tape resources for you. Use rsv to reserve tape devices before opening any tape files. The rsv utility does not allocate real tape devices, but it gives you permission to access tape devices.

The rsv utility accepts the following options:

-m *message_file* Specifies a file in which informative messages from the tape subsystem are written. The default *message_file* is tape.msg, which the system administrator can change. If you omit *message_file*, the name defined during installation of the tape subsystem by MSGFILE in tapedef.h is used. The tape subsystem uses this file until all the reserved resources are released. The tape subsystem always appends to this file.

-t Places the message file in the directory specified by TMPDIR. See the CAUTIONS section. If the directory specified by TMPDIR does not exist or cannot be written into, the message file is created or used in the current working directory.

resources Takes the form *resource* [*amount*]; *resource* specifies the device group name, and *amount* specifies the number of devices. You can repeat *resource* [*amount*]; see the following examples. If you omit *resource*, it defaults to an installation-specified device group name. If you omit *amount*, one device is assumed.

Before you issue the rsv utility, you must release all previously reserved resources. If the requested resource is not available, or if you have not released all previously reserved resources, no new reservation occurs.

NOTES

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

Privilege Text	Action
SMACDAC	Allowed to access any tape regardless of MAC and discretionary access control (DAC) restrictions. Device group MAC enforcement is still performed.
SDAC	Allowed to override DAC restrictions.

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

Active Category	Action
system, secadm	Allowed to read and write any tape, subject to security label restrictions on the device.
sysadm	Allowed to read and write any tape, subject to security label restrictions on the device and the tape.

If the PRIV_SU configuration option is enabled, root is allowed to override MAC and DAC restrictions. Device group MAC enforcement is still performed.

CAUTIONS

The file specified with the -m option or the default *message_file* in the current working directory is appended to and it may grow quite large.

The rsv utility creates a pipe in the directory defined as USER_DIR in tapedef.h; the default is $TMPDIR. If you change your $TMPDIR environment variable after a rsv utility, a tape request for tapeinfo fails.

EXIT STATUS

If rsv completes successfully, 0 is returned; otherwise, a nonzero value is returned. Where possible, this exit status code is normalized to the last three digits. Exit status values are documented in the *Tape Subsystem User's Guide*, Cray Research publication SG–2051.

EXAMPLES

The following examples illustrate different uses of the rsv utility.

Example 1: The rsv utility reserves one device named TAPE and one device named CART:

```
rsv TAPE 1 CART
```

Example 2: In the first command line of the example, rsv reads TAPE as a *resource* and 2 as the *amount*; it then reads CART as a *resource* and 1 as the *amount*. In the second command line, rsv reads TAPE as a *resource* and 1 as the *amount*; it then reads CART as the *resource*, and, because *amount* is not specified, one device is assumed.

```
rsv TAPE 2 CART 1
```

or

```
rsv TAPE 1 CART
```

Example 3: This example shows a usage of `rsv` that is not valid. In the first command line of the example, `rsv` reads `CART` as if it were an *amount*. In the second command line, `rsv` reads `2` as a *resource*.

```
rsv TAPE CART
```

or

```
rsv 2
```

Example 4: The `rsv` utility reserves one device of the default type, and it directs all messages to `msgfile` in your working directory:

```
rsv -m msgfile
```

If you move or remove the file specified by the `-m` option or the default *message_file* in the working directory before you are through processing tapes, messages will no longer be written.

FILES

`/usr/include/tapedef.h`	Definitions for trace file size

SEE ALSO

`privtext`(1), `rls`(1), `tpmnt`(1), `tprst`(1), `tpstat`(1)

General UNICOS System Administration, Cray Research publication SG–2301

Tape Subsystem User's Guide, Cray Research publication SG–2051

NAME

rusers – Lists names of users logged in on local machines (RPC version)

SYNOPSIS

/usr/bin/rusers [-ahilu] *hosts*

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The rusers utility produces output for remote machines that is similar to the output from the who(1) utility. It broadcasts on the local network and prints the responses it receives. (Cray Research systems are usually configured without connections to media that support broadcasting. Therefore, the rusers utility, without any hosts specified, may not return any user names.) Normally, the listing is in the order in which responses are received, but this order can be changed by the use of one of the options listed in this section.

The default is to print out a listing with one line per machine. When the -l flag is given, a who(1) style listing is used. If a user has not typed anything for a minute or more, the idle time is reported.

A remote host responds only if it is running the rusersd(8) daemon, which usually is started from inetd(8).

The rusers utility accepts the following options:

-a Displays a report for a machine even if no users are logged on.

-h Sorts alphabetically by host name.

-i Sorts by idle time.

-l Displays a longer listing in the style of who(1).

-u Sorts by number of users.

hosts When *host* arguments are given, rather than broadcasting, rusers will only query the list of specified hosts.

BUGS

Broadcasting does not work through gateways or on nonbroadcast media.

FILES

/etc/inetd.conf

SEE ALSO

`who`(1)

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

NAME

`sact` – Prints current SCCS file-editing activity

SYNOPSIS

`sact` *files*

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The `sact` utility informs the user of any impending deltas to a named Source Code Control System (SCCS) file. This situation occurs when `get`(1) with the `-e` option has been executed previously without a subsequent execution of `delta`(1). If a directory is named on the command line, `sact` behaves as though each file in the directory were specified as a named file, except that non-SCCS files and unreadable files are silently ignored. If a name of `-` is given, the standard input is read with each line being taken as the name of an SCCS file to be processed.

The output for each named file consists of five fields separated by spaces, as follows:

Field 1 Specifies the SID of a delta that currently exists in the SCCS file to which changes will be made to make the new delta

Field 2 Specifies the SID for the new delta to be created

Field 3 Contains the log name of the user who will make the delta (that is, executed a `get` for editing)

Field 4 Contains the date that `get -e` was executed

Field 5 Contains the time that `get -e` was executed

EXIT STATUS

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

0 Successful completion.

>0 An error occurred.

MESSAGES

Error messages from SCCS are printed. Use `help`(1) for explanations.

SEE ALSO

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

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

NAME

sag – Displays system activity graph

SYNOPSIS

sag [-s *time*] [-e *time*] [-i *sec*] [-f *file*] [-T *term*] [-x *spec*] [-y *spec*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The sag command graphically displays the system activity data stored in a binary data file by a previous sar(1) run. You can plot any of the sar(1) data items singly or in combination, as crossplots or versus time; simple arithmetic data combinations can be specified. The sag command invokes sar(1) and finds the desired data by string-matching the data column header (run sar to see what is available). The following options are passed to sar(1):

-s *time* Selects data later than *time* in the form *hh*[:*mm*]. The default is 08:00.

-e *time* Selects data up to *time*. The default is 18:00.

-i *sec* Selects data at intervals as close as possible to *sec* seconds.

-f *file* Uses *file* as the data source for sar(1). The default is the current daily data file /usr/adm/sa/sa*dd*.

The following options are also available:

-T *term* Produces output suitable for terminal *term*.

-x *spec* Produces x axis with *spec* in the form shown under the -y option.

-y *spec* Produces y axis with *spec* in the following form:

"*name*[*op name*]...[*lo hi*]"

The *name* variable is either a string that matches a column header in the sar report, with an optional device name in brackets (for example, reads[dsk-1]) or an integer value. *op* is the symbol +, -, ∗, or / surrounded by blanks. You can specify up to five names. sag does not recognize parentheses. Contrary to custom, + and - have precedence over ∗ and . Evaluation is left to right. Thus, A / A + B * 100 is evaluated (A/(A+B))∗100, and A + B / C + D is (A+B)/(C+D). *lo* and *hi* are optional numeric scale limits. If unspecified, they are deduced from the data.

A single *spec* is permitted for the x axis. If unspecified, *time* is used. Up to 5 *spec*s separated by semicolons (;) may be given for -y. Enclose the -x and -y arguments in double quotes ("") if blanks or \<CR> are included. The -y default is as follows:

```
-y "%usr 0 100; %usr + %sys 0 100; %usr + %sys + %wio 0 100"
```

BUGS

The `sag` command produces a command file consisting of graphic commands and several data files. These must be used on a machine that supports plots or graphs, such as a Sun Workstation. The files are transformed to graphs on a Sun workstation by using `tektool`, then executing the `sag`-produced command file, and piping the output through plot. The scaling produced by `sag` is not correct for Cray PVP systems.

EXAMPLES

Example 1: To see today's CPU usage, enter the following:

```
sag
```

Example 2: To see activity over 15 minutes of all disk drives, enter the following:

```
TS=`date +%H:%M`
sar -o tempfile 60 15
TE=`date +%H:%M`
sag -f tempfile -s $TS -e $TE -y "reads"
```

FILES

`/usr/adm/sa/sa`*dd* Daily data file for day *dd*

SEE ALSO

`sar`(1)

NAME

sar – Extracts operating system activity information

SYNOPSIS

sar [-a] [-b] [-c] [-d] [-g] [-h] [-j] [-k] [-l] [-n] [-o *file*] [-p] [-q] [-r] [-t] [-u] [-v] [-w] [-x] [-y] [-z] [-A] [-B] [-H] [-L] [-M] [-P] [-S] [-T] [-U] [-W] [-X] [-Z] *seconds* [*integral*]

sar [-a] [-b] [-c] [-d] [-e *time*] [-f *file*] [-g] [-h] [-i *sec*] [-j] [-k] [-l] [-n] [-p] [-q] [-r] [-s *time*] [-t] [-u] [-v] [-w] [-x] [-y] [-z] [-A] [-B] [-H] [-L] [-M] [-P] [-S] [-T] [-U] [-W] [-X] [-Z]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The sar command extracts operating system activity information according to a specified time interval. In the first instance, sar samples cumulative activity counters in the operating system. You specify the number of seconds between samples (*seconds*) and the total number of samples (*integral*). If you specify -o *file*, sar saves the samples in *file* in binary format.

In the second instance, sar extracts data from a previously recorded *file*, either the one specified by the -f option or, by default, the standard system activity daily data file, /usr/adm/sa/sa*dd*, for the current day *dd*. The starting and ending times of the report can be bounded through the -s *time* and -e *time* arguments by using the 24-hour clock form *hh*[:*mm*[:*ss*]]. The -i option selects records at *sec* second intervals; otherwise, all intervals found in the data file are reported.

In either case, you can specify subsets of data to be printed by using the following options (the fields reported are listed with each option):

-a	Reports use of file access system routines.	
	iget/s	Number of inode accesses per second.
	namei/s	Number of file path name lookups per second.
	dirblk/s	Number of file directory blocks read per second.
-b	Reports buffer activity.	
	bread/s, bwrit/s	Transfers per second of data between system buffers and disk or other block devices.
	lread/s, lwrit/s	Accesses of system buffers.
	%rcache, %wcache	Percentage of read/write buffer accesses that were statisfied by the buffer cache instead of having to go to a disk directly.
	pread/s, pwrit/s	Transfers through raw (physical) device mechanism.
-c	Reports system calls.	

`scall/s` System calls of all types.
`sread/s`, `swrit/s`, `fork/s`, `exec/s`
Specific system calls.
`rchar/s`, `wchar/s` Characters transferred by read and write system calls.

`-d` Reports activity for each disk device.

`reads writes` Number of data transfers from or to device during the interval. The number of bytes transferred can be computed by multiplying reads+writes by 4096.

On systems with an I/O subsystem model E (IOS-E):

`reads writes` Number of blocks transferred.
`ncyls` Number of cylinders crossed.
`avwait`, `avserv` Average time in milliseconds that transfer requests wait idly in the queue and average time to be serviced. This includes seeks, rotational latency, and data transfer times.
`rerrs`, `uerrs` Number of total recovered and unrecovered read and writer errors.

`-e` *time* Specifies the end time of report.

`-f` *file* Specifies the *file* from which `sar` extracts data.

`-g` Reports host kernel calls issued by a guest kernel. The information is displayed analogously to the `-t` option and includes the following:

`host call` The host call name.
`%time` Time spent by the host handling each call type as a percent of the interval total.
`calls/s` Number of calls per second.
`avetime` Average handling time (in microseconds).
`maxtime` Maximum handling time (in microseconds) across all reporting intervals.
`mintime` Minimum handling time (in microseconds) across all reporting intervals.

`-h` Reports terminal traffic and possible overflow of terminal (`clist`) buffers.

`-i` *sec* Selects records at *sec* intervals.

`-j` Reports process shuffles in memory and text/data locking and unlocking. Process shuffles occur when a `plock`(2) or `chmem`(2) system call is initiated or when real time mode is set. Text/data locking and unlocking occurs when `plock` is called.

`-k` Reports TCP/IP interrupt information. The system and the `sar` package must have been built with SCTRACE defined, and TCP/IP must be in the system.

`-l` Reports device cache (`ldcache`) activity.

`Cache to user: Reads`, `Writes`
Number of blocks read from cache to user and number of blocks written from user to cache.

`Cache to disk: Reads`, `Writes`
: Number of blocks read from disk to cache and number of blocks written from cache to disk.

`Cache/disk ratio: Read`, `Write`, `Total`
: Ratio of cache-to-user to cache-to-disk.

The `-l` option is only available to non-root users when `sar` is extracting data from a previously recorded file. It is not available to non-root users when `sar` is collecting data interactively from the system (when the seconds and integral parameters are specified), because the device from which these statistics are gathered is accessible only by root.

`-n` Reports network activity.

`network` Network address.

`ipkts/s`, `ierrs/s`, `opkts/s`, `oerrs/s`, `collis/s`
: Corresponds to input packets, input errors, output packets, output errors, and collisions, respectively.

`-o` *file* Saves the samples in *file* in binary format.

`-p` Reports CPU usage by processor.

`unix restarts: cpu`, `user`, `unix`, `idle`
: Portion of time running in CPU mode, user mode, running in the kernel, and otherwise idle, respectively.

`-q` Reports average queue length while occupied, and percentage of time occupied.

`runq-sz`, `%runocc` Run queue of processes in memory that are runnable.
`swpq-sz`, `%swpocc` Swap queue of processes swapped out, but ready to run.

`-r` Reports remote file (network file system (NFS)) activity.

`svcall/s` Server calls per second.
`%svread` Percent server reads.
`%svwrit` Percent server writes.
`%svother` Percent all others.
`clcall/s` Client calls per second.
`%clread` Percent client reads.
`%clwrite` Percent client writes.
`%clother` Percent all others.

`-s` *time* Specifies the start time of the report.

`-t` Reports system call information. This includes the system call name, percent of time spent in the call, number of calls per second, and the average, minimum, and maximum system call path lengths. Unused system calls during the period of time being reported are not shown. The system and the `sar` package must have been built with SCTRACE defined.

`-u` Reports CPU usage. The default report contains the following fields:

%usr	Portion of time running in user mode.
%sys	Portion of time running in system mode.
%wsem	Portion of time waiting on a semaphore.
#locks	Number of collisions for the system lock.
%idle	Time spent idle.
%wio	Time spent idle waiting for I/O.
%guest	Portion of time used by the guest(s). If sar is being run from a guest, the time reported is for the host.

%usr, %sys, %idle, and %guest make up the total time. %wsem is a component of %sys. %wio is a component of %idle.

-v Reports status of text, process, nc1inode, and file tables.

text-sz, proc-sz, nc1inod-sz, file-sz
: Entries/size for each table, evaluated once at sampling point.

text-ov, proc-ov, nc1inod-ov, file-ov
: Overflows occurring between sampling points.

-w Reports system swapping and switching activity.

swpin/s, swpot/s, bswin/s, bswot/s	Number of transfers and number of 4096-byte units transferred for swapins (including initial loading of some programs) and swapouts.
xswin/s, xswot/s	Number of times a shared-text process was swapped in and the number of times a shared-text segment was freed.
pswch/s	Process switches.

-x Reports IOS packets into the Cray Research system and out to the IOS.

-y Reports tty device activity.

rawch/s, canch/s, outch/s	Input character rate, input character rate processed by canon, and output character rate.
rcvin/s, xmtin/s	T-packets received and transmitted (a t-packet is a terminal packet type).

-z Reports asynchronous I/O usage; in other words, sysrda and syswra report buffer cache asynchronous usage. aread, awrite, and listio report usage of reada, writea, and listio system calls.

-A Reports all data. Equivalent to specifying -abcdhjklpqtuvwxyzBHMTXWZ.

-B Produces an expanded version of the -b report. In addition to the information found in the -b report, the following additional information is given:

brblks, bwblks	Number of blocks transferred (read/written) between system buffers and disk or other block devices.
lrblks, lwblks	Number of blocks moved from/to system buffers.
prblks, pwblks	Number of blocks read/written via raw (physical) I/O.

`-H` System call history option. Reports system call information for the time period from boot to the present. This information includes the system call name, percent of time spent in the call, number of calls per second, and the average, minimum, and maximum system call path lengths. Unused system calls during the period being reported are not shown. The system and the `sar` package must have been built with SCTRACE defined.

`-L` Reports kernel multi-threaded lock statistics.

`Lid`	Kernel lock identifier.
`locks/s`	Number of lock attempts per second.
`avethrd`	Average kernel thread time in microseconds while locked.
`hold/s`	Number of lock attempts per second that resulted in a hold to wait while another CPU had the lock.
`avehold`	Average hold time in microseconds.
`%wsem`	Percent of wait time this kernel lock accounted for.

NOTE: For data to appear in the `locks/s` and `avethrd` fields, the kernel must have been built using the `SEMTIMING` option. By default, this is not done, due to the significant kernel overhead that it adds.

`-M` Reports memory and swap usage.

`umemtot`	Amount of memory available for use by user processes.
`umemuse`	Amount of user memory in use.
`memlock`	Amount of locked memory.
`swaptot`	Total amount of swap space.
`swapuse`	Amount of swap space in use.

`-P` Reports the same data as the `-p` option, except the `unix` field has been expanded into two fields.

`unixc`	Percent of time running in system mode executing system calls on behalf of users.
`unixk`	Percent of time running in system mode executing kernel functions on behalf of the system. This includes interrupt processing time and semaphore wait time.

`-S` Summary format. Reports raw totals from the last `unix restart` or from a specified interval. Valid only for the `-b`, `-d`, `-n`, `-r`, `-t`, `-u`, `-w`, and `-L` options.

`-T` Reports activity for each tape drive.

`mounts`	Number of volumes mounted.
`reads writes`	Number of data transfers from or to the device during the interval. The number of bytes transferred can be computed by multiplying each value by 4096.

`-U` Reports the same data as the `-u` option, except the `sys` field has been expanded into two fields, not including semaphore wait time.

`sysc`	Percent of time running in system mode executing system calls on behalf of users.

`sysk` Percent of time running in system mode executing kernel functions on behalf of the system. This includes interrupt processing time.

`-W` Reports the number of times per second that processes are loaded and runnable, but not running, and the average number of processes.

`-X` Reports the number of abnormal exchanges from user programs, by CPU (abnormal exchanges = err, fpi, ore, pre, dli).

`-Z` Reports the same data as the `-z` option, plus the number of blocks transferred by `aread` and `awrite`.

seconds Specifies the number of seconds between samples.

[*integral*] Specifies the total number of samples.

CAUTIONS

When collecting `sar` data over short intervals (a few seconds), it is possible for `sar` to report values that appear inconsistent. This is true when using `sar` to collect the data or by calling `sadc` directly (see `sar`(8)). For example, the percentage displayed in the `%swpocc` field is more than 100%. This occurs because not all the data values used by `sar` are updated at exactly the same rate. In addition, `sadc` samples data in several steps, which can lead to additional inaccuracies if updates occur between sampling steps. Consider the `%swpocc` field. This field is calculated by subtracting the old data from the new data and dividing by time; however, since the data is only updated 1 time per second by the kernel, it is likely that data samples taken at a 1-second rate will have percentage values greater than 100%.

EXAMPLES

Example 1: The following example shows today's CPU activity so far. Data is extracted from the file `/usr/adm/sa/sa`*dd*.

```
$ sar
```

Example 2: The following example collects three samples that contain data about all system activity. Data samples are written to the `tmp` file every 20 minutes.

```
$ sar -o temp 1200 3
```

Example 3: The following example reports disk and tape activity for data collected in example 2.

```
$ sar -dT -f temp
```

FILES

`/usr/adm/sa/sa`*dd* Daily data file, where *dd* are digits representing the day of the month

SEE ALSO

`sag`(1)

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

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

NAME

scanit – Corrects code for certain user programs on CRAY J90 systems and CRAY EL98 systems

SYNOPSIS

scanit [-e *feature*] [-o *output_file*] [-v] *input_file*

IMPLEMENTATION

Cray PVP systems

Necessary for multitasking code executing on CRAY EL98 systems and code that enables scalar cache on CRAY J90 systems.

DESCRIPTION

Hardware instruction sequences have been discovered that can generate unexpected results in certain situations. segldr(1) and the scanit command render an executable file (a.out) safe in the following situations:

- When it uses multitasking on CRAY EL98 systems
- When it enables cache memory on CRAY J90 systems

If your program does not fall into one of these categories, or if you load your program with SEGLDR version 8.0.4 or later, you have no reason to use scanit. The segldr(1) command and scanit provide the same safety features.

scanit first analyzes and then modifies your a.out file. It addresses only the two issues it was designed to make safe; it will not affect the program's performance significantly or change the answers you receive.

scanit accepts the following options:

-e *feature* Analyzes and makes safe the program that makes use of *feature*. The choices for *feature* are as follows:

- cache — Makes *output_file* safe for using cache on CRAY J90 systems.
- tasking — Makes *output_file* safe for multitasking on CRAY EL98 systems.

-o *output_file* Names the file output by scanit. The default name is a.out; however, scanit will not overwrite *input_file*, so ensure that the two names do not match.

-v Prints additional analysis information: specifically, the number of instruction sequences containing a possible hazard. Use this information to determine whether a hazard condition exists.

input_file Names the executable program to be analyzed and made safe.

ENVIRONMENT VARIABLES

If you set the TARGET environment variable to identify the machine characteristics of the system on which the program will execute, scanit will use that information. If you do not use the TARGET variable, scanit assumes the host system. The use of the -e *feature* option overrides the TARGET environment variable or host system machine characteristics.

NOTES

You are strongly urged to reload your CRAY EL98 and CRAY J90 programs with SEGLDR version 8.0.4 or later. The scanit command should only be used when reloading with segldr(1) is impractical. While the behavior of scanit mimics that of segldr(1), scanit operates on an existing executable program rather than .o files and libraries. segldr(1) contains no new options to make programs safe; it performs the analysis and any corrections automatically, based on the TARGET information for the host system.

In rare circumstances, you may be unable to reload a program with segldr(1). For instance, this is the case when the source files for a program are not available because the program is distributed in executable form by a vendor other than Cray Research. If you have an agreement with another vendor that does not allow you to modify executable code, do not use scanit.

When the scanit command produces an executable program that can safely use cache on a CRAY J90 system, it sets a bit in the output a.out file to indicate to the UNICOS kernel that cache should be enabled.

Be aware that the scanit tool may have difficulty finding enough fixup space in segmented programs if the segments are small. Relinking segmented codes with SEGLDR version 8.0.4 or later should work, because it adds extra code space in non-root segments.

EXIT STATUS

The scanit command exits with one of the following values:

0 Successful completion. The output file can be safely executed.

1 The scanit command was unable to make the output file safe. A warning message to this effect will be printed to stderr. In addition, if run it on a CRAY J90 system, the program will run without cache enabled (the enable-cache bit will not be set). On a CRAY EL98 system, the permission bits to enable execution will not be set. If you change the permission bits and run the program on a CRAY EL98 system, you may get error conditions or incorrect answers; the program will run correctly on other Cray PVP systems.

EXAMPLES

Example 1: In the following example, all of the defaults are used. scanit uses TARGET information to determine whether the host is a CRAY J90 or a CRAY EL98 system (that is, whether to fix cache access or multitasking problems) and writes the output to a.out.

```
scanit my.code
exec a.out
```

Example 2: Use the following command line on a CRAY EL98 multitasking executable named `a.out`:

```
scanit -e tasking -o safe.out a.out
exec safe.out
```

Example 3: Use the following command line to access cache memory on a CRAY J90 system. This example takes an executable file named `program.output` and, because the `-o` option is missing, writes the output to `a.out`, the default. The `-v` option prints the number of problems corrected.

```
scanit -v -e cache program.output
File a.out contains fixes for 628 cache hazard conditions

exec a.out
```

If there had been no problems with the executable file, the `-v` option would have printed the following:

```
No hazard conditions were found in file program.output
```

SEE ALSO

`segldr`(1)

NAME

sccs – Front end for the SCCS subsystem

SYNOPSIS

sccs [-r] [-d *path*] [-p *path*] *command* [*options...*] [*operands...*]

IMPLEMENTATION

All Cray Research systems

STANDARDS

XPG4

DESCRIPTION

The sccs utility is a front end to the SCCS programs. It also includes the capability to run set-user-id to another user to provide additional protection.

The sccs utility invokes the specified *command* with the specified *options* and *operands*. By default, each of the *operands* is modified by prefixing it with the string SCCS/s.

The *command* operand can be one of the SCCS utilities in this document (admin, delta, get, prs, rmdel, sact, unget, val, or what) or one of the pseudo-utilities listed in the DESCRIPTION section.

The sccs utility accepts the following options, except that *options* operands are actually options to be passed to the utility named by *command*. When the portion of the command:

command [*options...*] [*operands...*]

is considered, all of the pseudo-utilities used as *command* support the Utility Syntax Guidelines. Any of the other SCCS utilities that can be invoked in this manner support the Guidelines to the extent indicated by their individual OPTIONS sections.

The following options are supported preceding the *command* operand:

-d *path* A path name of a directory to be used as a root directory for the SCCS files. The default is the current directory. The -d option takes precedence over the PROJECTDIR variable. See -p.

-p *path* A path name of a directory in which the SCCS files are located. The default is the SCCS directory.

The -p options differs from the -d option in that the -d option-argument is prefixed to the entire path name and the -p option-argument is inserted before the final component of the path name. For example:

```
sccs -d /x -p y get a/b
```

will convert to:

```
get /x/a/y/s.b
```

This allows the creation of aliases such as:

```
alias syssccs="sccs -d /usr/src"
```

that will be used as:

```
syssccs get cmd/who.c
```

`-r` Invokes *command* with the real user ID of the process, not any effective user ID that the `sccs` utility is set to. Certain commands (`admin`, `check`, `clean`, `diffs`, `info`, `rmdel`, and `tell`) cannot be run set-user-ID by all users, since this would allow anyone to change the authorizations. These commands are always run as the real user.

The `sccs` utility accepts the following operands:

command An SCCS utility name or the name of one of the pseudo-utilities listed in the DESCRIPTION section.

options An option or option-argument to be passed to *command*.

operands An operand to be passed to *command*.

The following environment variables affect the execution of `sccs`:

`LANG` Provides a default value for the internationalization variables that are unset or null. If `LANG` is unset or null, the corresponding value from the implementation-specific default locale will be used. If any of the internationalization variables contains an invalid setting, the utility will behave as if none of the variables had been defined.

`LC_ALL` If set to a nonempty string value, override the values of all the other internationalization variables.

`LC_CTYPE` Determines the locale for the interpretation of sequences of bytes of text data as characters (for example, single- as opposed to multibyte characters in arguments and input files).

`LC_MESSAGES` Determines the locale that should be used to affect the format and contents of diagnostic messages written to standard error.

`NLSPATH` Determines the location of message catalogs for the processing of `LC_MESSAGES`.

`PROJECTDIR` Provides a default value for the `-d` *path* option. If the value of `PROJECTDIR` begins with a slash, it is considered an obsolete path name; otherwise, the home directory of a user of that name is examined for a subdirectory `src` or `source`. If such a directory is found, it is used. Otherwise, the value is used as a relative path name.

Additional environment variable effects may be found in the utility description for the specified *command*.

Many of the SCCS utilities take directory names as operands as well as specific filenames. The pseudo-utilities supported by `sccs` are not described as having this capability, but are not prohibited from doing so.

The following pseudo-utilities are supported as *command* operands. All options referred to in the following list are values given in the *options* operands following *command*.

`check` Equivalent to `info`, except that nothing is printed if nothing is being edited, and a non-zero exit status is returned if anything is being edited. The intent is to have this included in an "install" entry in a makefile to ensure that everything is included into the SCCS file before a version is installed.

`clean` Removes everything from the current directory that can be recreated from SCCS files, but do not remove any files being edited. If the `-b` option is given, branches are ignored in the determination of whether they are being edited; this is dangerous if branches are kept in the same directory.

`create` Creates an SCCS file, taking the initial contents from the file of the same name. Any options to `admin` are accepted. If the creation is successful, the original files are renamed by prefixing the basenames with a comma. These renamed files should be removed after it has been verified that the SCCS files have been created successfully.

`delget` Performs a `delta` on the named files and then `get` new versions. The new versions will have ID keywords expanded and will not be editable. Any `-m`, `-p`, `-r`, `-s`, and `-y` options will be passed to `delta`, and any `-b`, `-c`, `-e`, `-i`, `-k`, `-l`, `-s`, and `-x` options will be passed to `get`.

`deledit` Equivalent to `delget`, except that the `get` phase includes the `-e` option. This option is useful for making a checkpoint of the current editing phase. The same options will be passed to `delta` as described above, and all the options listed for `get` above except `-e` are passed to `edit`.

`diffs` Writes a difference listing between the current version of the files checked out for editing and the versions in SCCS format. Any `-r`, `-c`, `-i`, `-x`, and `-t` options are passed to `get`; any `-l`, `-s`, `-e`, `-f`, `-h`, and `-b` options are passed to `diff`. A `-C` option is passed to `diff` as `-c`.

`edit` Equivalent to `get -e`.

`fix` Removes the named delta, but leaves a copy of the delta with the changes that were in it. It is useful for fixing small compiler bugs, and so forth. It must be followed by a `-r` *SID* option. Since `fix` does not leave audit trails, it should be used carefully.

`info` Writes a listing of all files being edited. If the `-b` option is given, branches (that is, SIDs with two or fewer components) are ignored. If a `-u` *user* option is given, then only files being edited by the named user are listed. A `-U` option is equivalent to `-u` <*current user*>.

`print` Writes out verbose information about the named files, equivalent to `sccs prs`.

`print` Writes a newline-separated list of the files being edited to standard output. Takes the `-b`, `-u`, and `-U` options like `info` and `check`.

`unedit` This is the opposite of an `edit` or a `get -e`. It should be used with caution, since any changes made since the `get` will be lost.

EXIT STATUS

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

0 Successful completion.

>0 An error occurred.

EXAMPLES

Example 1: To get a file for editing, edit it and produce a new delta:

```
sccs get -e file.c
ex file.c
sccs delta file.c
```

Example 2: To get a file from another directory:

```
sccs -p /usr/src/sccs/s. get cc.c
```

or

```
sccs get /usr/src/sccs/s.cc.c
```

Example 3: To make a delta of a large number of files in the current directory:

```
sccs delta *.c
```

Example 4: To get a list of files being edited that are not on branches:

```
sccs info -b
```

Example 5: To delta everything being edited by the current user:

```
sccs delta $(sccs tell -U)
```

Example 6: In a makefile, to get source files from an SCCS file if it does not already exist:

```
SRCS = <list of source files>
$(SRCS):
       sccs get $(REL) $@
```

NAME

`sccsdiff` – Compares two versions of an SCCS file

SYNOPSIS

`sccsdiff -r`*SID1* `-r`*SID2* [`-p`] [`-s`*n*] *files*

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `sccsdiff` utility compares two versions of a Source Code Control System (SCCS) file and generates the differences between the two versions. Any number of SCCS files may be specified; the arguments apply to all files.

The `sccsdiff` utility accepts the following options and arguments:

`-r`*SID#* *SID1* and *SID2* specify the deltas of an SCCS file that are to be compared. Versions are passed to `bdiff`(1) in the order given.

`-p` Pipes output for each file through `pr`(1).

`-s`*n* *n* is the file segment size that `bdiff`(1) passes to `diff`(1). This is useful when `diff`(1) fails because of a high system load.

files Specifies the SCCS files to be compared.

MESSAGES

`file: No differences` The two versions are the same.

Error messages from SCCS are printed. Use `help`(1) for explanations.

EXAMPLES

The differences between delta `1.1` and `1.2` in file `s.example.c` are written to `stdout`. User input is shown in bold type:

```
$ sccsdiff -r1.1 -r1.2 s.example.c
4c4
<       printf("Hello, world\n");
---
>       printf("Hello, world!\n");
$
```

FILES

`/tmp/get`*?????* Temporary files

SEE ALSO

`admin`(1), `bdiff`(1), `cdc`(1), `comb`(1), `delta`(1), `diff`(1), `get`(1), `help`(1), `pr`(1), `prs`(1), `rmdel`(1), `sact`(1), `unget`(1), `val`(1), `vc`(1), `what`(1)

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

NAME

`script` – Makes a typescript of a terminal session

SYNOPSIS

`script` [`-a`] [`-k`] [`-n`] [`-q`] [`-s`] [`-S` *shell*] [*file*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `script` utility saves characters written to your terminal in a file. If you do not specify a *file* name, the characters are saved in a file called `typescript`.

The script ends when the forked *shell* exits.

This program is useful when you are using a CRT and want a copy of the dialog.

The `-k`, `-n`, `-s`, and `-S` options control which shell is used. If these options are not specified, `script` will attempt to determine the correct shell from the environment.

The `script` utility accepts the following options:

`-a` Appends to the `typescript` file instead of creating a new file.

`-k` Invokes `/bin/sh`.

`-n` Invokes `/bin/csh`.

`-q` Invokes `quiet mode`, in which the `script started` and `script done` messages are turned off.

`-s` Invokes `/bin/sh`.

`-S` *shell* Lets you specify the shell.

file File that collects characters written to your controlling terminal. The default depends on your `SHELL` environment variable.

SEE ALSO

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

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

NAME

`sdiff` – Compares programs side-by-side

SYNOPSIS

`sdiff` [`-l`] [`-o` *output*] [`-s`] [`-w` *n*] *file1 file2*

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `sdiff` utility uses the output of `diff`(1) to produce a side-by-side listing of two files, indicating the lines that are different. Each line of the two files is printed with a blank gutter between them if the lines are identical, a `<` in the gutter if the line exists only in *file1*, a `>` symbol in the gutter if the line exists only in *file2*, and a `|` symbol for lines that are different.

Example:

```
x       |       y
a               a
b       <
c       <
d               d
        >       c
```

The `sdiff` utility accepts the following options:

`-l` Prints only the left side of any lines that are identical.

`-o` *output* Uses the next argument, *output*, as the name of a third file that is created as a user-controlled merging of *file1* and *file2*. Identical lines of *file1* and *file2* are copied to *output*. Sets of differences, as produced by `diff`(1), are printed; where a set of differences share a common gutter character. After printing each set of differences, `sdiff` prompts the user with % and waits for one of the following user-typed commands:

- `l` Appends the left column to the output file.
- `r` Appends the right column to the output file.
- `s` Turns on silent mode; does not print identical lines.
- `v` Turns off silent mode.
- `e l` Calls the editor with the left column.
- `e r` Calls the editor with the right column.
- `e b` Calls the editor with the concatenation of left and right.
- `e` Calls the editor with a zero-length file.

`q` Exits from the program.

On exit from the editor, the resulting file is concatenated on the end of *output.*

`-s` Does not print identical lines.

`-w` *n* Uses the next argument, *n*, as the width of the output line. The default line length is 130 characters.

file1, *file2* Files to be compared.

NOTES

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

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

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

CAUTIONS

Lines from input files which exceed `sdiff`'s internal buffer are truncated.

SEE ALSO

`diff`(1), `ed`(1)

NAME

sdss – Reports status information about the secondary data segment pool

SYNOPSIS

sdss [-a] [-c] [-d] [-e] [-l] [-r] [-s] [-w] [-j *jidlist*] [-p *pidlist*] [-u *uidlist*]

IMPLEMENTATION

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

DESCRIPTION

The sdss command prints information about processes and logical device caches by using the secondary data segments (SDS) pool. The following options control the information display:

-a Prints the base address of the SDS. Must be accompanied by the -c or -d option.

-c Prints information about logical device cache segments.

-d Prints information about all processes with existing SDS.

-e Implies all other options.

-l Prints information about SDS limits. Implies the -d option.

-r Raw mode; suppresses the header and summary lines. This is useful for piping the output into sort(1).

-s Prints the login name, rather than the numerical user ID. Implies the -d option.

-w Prints information about processes waiting on an ssbreak(2) system call. Implies the -d option.

-j *jidlist* Restricts the listing to data about processes that have job ID numbers in *jidlist*. Implies the -d option.

-p *pidlist* Restricts the listing to data about processes that have process ID numbers in *pidlist*. Implies the -d option.

-u *uidlist* Lists only data about processes that have a user ID number or login name in *uidlist*. Implies the -d option.

If you do not specify any options, an information summary is given detailing basic usage statistics.

Definitions for the column headings in an sdss listing follow. The letters under the option heading indicate the options that cause the corresponding heading to appear. Note that a ! symbol preceding an option letter indicates that the option has not been specified.

Heading	Option	Description
ADDR	-a	Base address of the segment relative to the base address of the SDS pool.
SIZE	-c,-d	Size of the segment in SDS units (a unit is 4096 bytes).
SBRK	-d,-w	Size of the ssbreak increment for which the process is waiting.
LIM	-d,-l	Maximum SDS size allowed to the process. A letter J appended to the value indicates a per job limit, and a letter P appended to the value indicates a per process limit.
PID	-d	Process ID of the process; you can kill or checkpoint a process if you know this number.
JID	-d	Job ID of the process; you can checkpoint a job if you know this number.
UID	-d	User ID number of the process owner; the login name is printed under the -s option.
F	!-l,!-w	Flags (octal and additive) associated with the process: 01 In core 02 System process 04 Locked in core (as for physical I/O) 10 Being swapped 20 Being traced by another process 100 Connected to CPU 200 Suspended for single threading 2000 Suspended for deadlock 4000 Suspended by user 10000 CPU limit exceeded 20000 Recoverable process 40000 Selecting 100000 Idle process 200000 Suspend in process 400000 Another tracing flag

Heading	Option	Description
S	!-l,!-w	The state of the process: S Sleeping W Waiting R Running 0 Running, connected to CPU 0 1 Running, connected to CPU 1 2 Running, connected to CPU 2 3 Running, connected to CPU 3 I Intermediate Z Terminated T Stopped X Growing
PRI	!-l&!-w	Priority of the process; higher numbers mean lower priority.
NI	!-l&!-w	Nice value; used in priority computation.
TIME	-d	Cumulative execution time for the process.
CMD/LDCACHE	-c,d	Command name or the logical device cache name.

NOTES

Output from sdss is restricted to processes running at a security label that the calling user dominates.

If this command is installed with the default privilege assignment list (PAL), a user with the showall privilege text is not subject to output restrictions.

BUGS

The status of the SDS pool can change while sdss is running; the picture it gives is only a close approximation to reality.

SEE ALSO

privtext(1), ps(1)

General UNICOS System Administration, Cray Research publication SG–2301

NAME

sed – Invokes the stream editor

SYNOPSIS

sed [-g] [-n] *script* [*files*]
sed [-g] [-n] [-e *script*]... [-f *sfile*]... [*files*]

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4
AT&T extension (-g option)

DESCRIPTION

The sed utility copies *files* (standard input by default) to standard output, edited according to a script of commands. The script is obtained from either the *script* operand string or a combination of the arguments from the -e *script* and -f *sfile* options.

The sed utility accepts the following options:

-e *script* Adds the editing commands specified by the *script* argument to the end of the script of editing commands. The *script* argument has the same properties as the *script* operand.

-f *sfile* Adds the editing commands from *sfile* to the end of the script.

-g For every substitute (s) command, performs a global replacement.

-n Suppresses the default output. Only lines explicitly selected for output are written.

files Files to be copied.

The -f option causes the script to be taken from file *sfile*; these options accumulate. If only one -e option and no -f options are specified, you can omit the -e flag.

A script consists of editing commands, one per line, of the following form:

[*address* [, *address*]] *function* [*arguments*]

In normal operation, sed cyclically copies a line of input, less its terminating <newline>, into pattern space (unless something is left after a D command), applies in sequence all commands whose addresses select pattern space, and at the end of the script, copies the pattern space to the standard output (except under -n) and deletes the pattern space.

Some of the commands use hold space to save all or part of pattern space for subsequent retrieval.

address is a decimal number that counts input lines cumulatively across files, a `$` that addresses the last line of input, or a context address, that is, a */ regular\xpression /* in the style of `ed`(1) and modified as follows:

- In a context address, the construction `\?`*regular expression*`?`, in which *?* is any character, is identical to */ regular expression /*. In the context address `\xabc\xdefx`, the second `x` stands for itself; therefore, the regular expression is `abcxdef`.
- Escape sequence `\n` matches a `<newline>` character embedded in the pattern space.
- A period `.` matches any character except the terminal `<newline>` character of the pattern space.
- A command line with no addresses selects every pattern space.
- A command line with one address selects each pattern space that matches the address.
- A command line that has two addresses selects the inclusive range from the first pattern space that matches the first address through the next pattern space that matches the second. (If the second address is a number less than or equal to the line number first selected, only one line is selected.) Thereafter, the process is repeated, looking again for the first address.

Editing commands can be applied only to nonselected pattern spaces by using the negation function `!` described in this section.

In the following list of functions, parentheses enclose the maximum number of permissible addresses for each function.

The *text* argument consists of one or more lines, all but the last of which end with `\` to hide the `<newline>`. Backslashes in text are treated like backslashes in the replacement string of an `s` command, and may be used to protect initial `<blank>`s and `<tab>`s against the stripping that is done on every script line. The *rfile* or *wfile* argument must terminate the command line and must be preceded by one or more `<blank>`s. Each *wfile* is created before processing begins. There can be at most 10 distinct *wfile* arguments.

(1)`a\`
text Appends. Place *text* on the output before reading the next input line.

(2)`b` *label* Branches to the `:` command bearing the *label*. If *label* is empty, branch to the end of the script.

(2)`c\`
text Changes. Deletes the pattern space. With zero or one address or at the end of a two-address range, place *text* on the output. Start the next cycle.

(2)`d` Deletes the pattern space. Starts the next cycle.

(2)`D` Deletes the initial segment of the pattern space through the first `<newline>`. Starts the next cycle.

(2)`g` Replaces the contents of the pattern space by the contents of the hold space.

(2)`G` Appends the contents of the hold space to the pattern space.

(2)h Replaces the contents of the hold space by the contents of the pattern space.

(2)H Appends the contents of the pattern space to the hold space.

(1)i\
text Insert. Place *text* on the standard output.

(2)l Lists the pattern space on the standard output in an unambiguous form. Nonprinting characters are written in 2-digit ASCII, and long lines are folded. (Standard escapes (see the following table) are used for things such as <tab> and <form-feed>.)

007	\a	<alert>
010	\b	backspace
011	\t	<tab>
012	\n	<newline>†
013	\v	<vertical-tab>
014	\f	<form-feed>
015	\r	<carriage-return>

† The sed utility cannot produce this.

(2)n Copies the pattern space to the standard output. Replace the pattern space with the next line of input.

(2)N Appends the next line of input to the pattern space with an embedded <newline>. (The current line number changes.)

(2)p Print. Copies the pattern space to the standard output.

(2)P Copies the initial segment of the pattern space through the first <newline> to the standard output.

(1)q Quit. Branches to the end of the script. Do not start a new cycle.

(2)r *rfile* Reads the contents of *rfile*. Place them on the output before reading the next input line.

(2)s/*regular expression*/*replacement*/*flags*
Substitutes the *replacement* string for instances of the *regular expression* in the pattern space. You can use any character other than backslash or <newline> instead of /. For a complete description, see ed(1). *flags* is zero or more of the following:

n *n*=1-512. Substitutes for just the *n*-th occurrence of the *regular expression.*

g Global. Substitutes for all nonoverlapping instances of the *regular expression* rather than just the first one.

p Prints the pattern space if a replacement was made.

w *wfile* Write. Appends the pattern space to *wfile* if a replacement was made.

(2)t *label* Test. Branches to the : command that bears *label* if any substitutions were made since the most recent reading of an input line or execution of a t. If *label* is empty, branch to the end of the script.

(2)`w` *wfile* Write. Appends the pattern space to *wfile*.

(2)`x` Exchanges the contents of the pattern and hold spaces.

(2)`y`/*string1*/*string2*/
Transform. Replaces all occurrences of characters in *string1* with the corresponding character in *string2*. The lengths of *string1* and *string2* must be equal.

(2)`!` *function* Negation. Applies *function* (or group, if *function* is `{`) only to lines not selected by the address(es).

(0)`:` *label* This command does nothing; it bears a *label* to which `b` and `t` commands can branch.

(1)`=` Places the current line number on the standard output as a line.

(2)`{` Executes the following `sed` commands through a matching `}` only when the pattern space is selected. The list of `sed` commands are separated by `<newline>`s. The `{` can be preceded with `<blank>`s and can be followed with white space. The *commands* may be preceded with white space. The terminating `}` must be preceded by a `<newline>` and then zero or more `<blank>`s.

(0) An empty command is ignored.

(0)# If # appears as the first character on the first line of a script file, that entire line will be treated as a comment, except if the character after the # is `n`, in which case the default output is suppressed. The rest of the line after #`n` is also ignored. A script file must contain at least one noncomment line.

NOTES

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

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

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

EXIT STATUS

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

0 Successful completion.

>0 An error occurred.

SEE ALSO

awk(1), ed(1), grep(1)

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

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

The UNIX Programming Environment, Brian W. Kernighan and Rob Pike, Prentice-Hall, Inc., 1984

NAME

segldr – Invokes the Cray Research segment loader (SEGLDR)

SYNOPSIS

segldr [-A *file*] [-a] [-b *value*] [-D *dirstring*] [-E] [-e *name*] [-F] [-f *value*] [-g] [-H *hi*[+*he*]] [-i *dirfiles*] [-j *names*] [-k] [-L *ldirs*] [-l *names*] [-M *arguments*] [-m] [-N] [-n] [-O *keyword*] [-o *outfile*] [-S *si*[+*se*]] [-s] [-t] [-u *unames*] [-V] [-z *file*] [-Z] *files*

IMPLEMENTATION

Cray PVP systems

DESCRIPTION

The segldr utility links relocatable object modules to produce an executable program. Load maps, if selected, are written to the stdout file by default (see the -M option). Error messages are written to the stderr file by default (see the -k option). The output file is execute-enabled when no warning or fatal errors occur during the load.

The segldr utility accepts the following options:

-A *file* Specifies executable file containing symbol information for SEGLDR. Symbol references in newly loaded code are linked to addresses in the existing executable code. The resultant output file then contains a code fragment that can execute in the existing program address space.

-a Aligns all code and local data blocks on instruction buffer boundaries.

-b *value* Adds 1024 times *value* number of words to the BSS (uninitialized data) area of the loaded program.

-D *dirstring* Passes *dirstring* of SEGLDR directives to SEGLDR. The *dirstring* argument is a character string of global SEGLDR directives separated with semicolons.

-E Echoes all processed directives to the load map file. See the -M option.

-e *ename* Sets the program entry address to the value of symbol *ename*.

-F Loads all modules from bin files, whether or not they are referenced.

-f *value* Fills uninitialized, statically allocated areas of the program with *value*. The *value* argument may be one of the following:

- zeros Fills with 0 bits (default).
- ones Fills with 1 bits.
- indef Fills with 0605054000000000000000 octal, to cause a floating-point error if referenced.

`-indef` Fills with 1605054000000000000000 octal, to cause a floating-point error if referenced.

`indefa` Sets uninitialized data to the product of a logical OR operation of O'0605054000000000000000 multiplied by the address of the word being preset. This value is the same as that of `indef`, except that the address of the word referenced will appear in the low-order bits of the value.

`-indefa` Sets uninitialized data to the product of a logical OR operation of O'1605054000000000000000 multiplied by the address of the word being preset. This value is the same as that of `-indef`, except that the address of the word referenced will appear in the low-order bits of the value.

A 16-bit octal value
Stores the value in each parcel of each uninitialized word.

`-g` Generates the Debug Symbol tables and appends them to the executable file. This option is enabled by default (see the `-s` option).

`-H` *hi*[+*he*] Assigns initial heap size (*hi*) and heap expansion increment (*he*). Specify sizes in words.

`-i` *dirfiles* Reads and processes the directives in the directives files. *dirfiles* contains a list of directives file names, separated by commas. When a `-` is present as one of the file names, the `segldr` utility reads the `stdin` file for directives. When a name begins with a `.` or `/` symbol, the loader assumes it is a complete path name and uses it without modification. Otherwise, the loader checks for the named files in the current directory.

`-j` *names* Reads and processes the directives in the directives files. *names* contains a list of directives file names, separated by commas. When a name begins with a `.` or `/` symbol, the loader assumes it is a complete path name and uses it without modification. Otherwise, the loader checks for a `segdir/`*name* file in the list of search directories and uses the first one found. See the `-L` option for the list of search directories.

`-k` Redirects all but summary-class error messages to the load map file. See the `-M` option.

`-L` *ldirs* Changes the `-l` option search algorithm to look for library files in directories *ldirs* before looking in the `/opt/ctl/craylibs/craylibs`, `/lib`, or `/usr/lib` directories. If the `-F` option is used to include the system default directories, the loader searches directories *ldirs* for those libraries before searching the `/opt/ctl/craylibs/craylibs`, `/lib`, or `/usr/lib` directories. Multiple `-L` options are cumulative.

`-l` *names* Identifies library files. When a name begins with a `.` or `/` symbol, it is assumed to be a full path name, and the `segldr` utility uses it as is. Otherwise, the `segldr` utility checks first for files `/opt/ctl/craylibs/craylibs/lib`*name*`.a` and `/lib/lib`*name*`.a`, and then for file `/usr/lib/lib`*name*`.a`. It uses the first one found. See the `-L` option.

`-M` *file* or `-M`, *opts* or `-M` *file,opts*
Selects an optional load map file, *file*, and the type of map to produce. If *file* is present, the `segldr` utility writes the load maps to that file in a paginated format, 132 characters per line. If a file is not provided, the `segldr` utility writes the load maps to the `stdout` file in a nonpaginated format, 80 characters per line. Load map options (*opts*) are as follows:

`s` or `stat`	Lists only load statistics
`a` or `address`	Sorts block map by address (the default map, if no *opt* is specified)
`al` or `alpha`	Sorts block map by name
`b` or `brief`	Restricts maps to `bin` files only
`c` or `cbxrf`	Lists common-block cross-references
`e` or `epxrf`	Lists entry-point cross-references
`p` or `part`	Lists a combination of `address` and `alpha`
`f` or `full`	Lists all load maps

`-m` Generates the address-level load map and writes it to the `stdout` file. This option is equivalent to the `-M`, `address` option.

`-N` Inhibits the inclusion of the default libraries in the load.

`-n` Generates a shared-text program.

`-O` *keyword* Selects allocation order. The *keyword* variable can be the following:

`tdb`	Allocates all code, followed by all initialized data, followed by all uninitialized data

`-o` *outfile* Writes executable program to the *outfile* file. When the `-o` option is not used, the executable program is written to the file named by the `ABS` directive. When neither the `-o` option nor `ABS` is specified, the executable output is written to file `a.out`.

`-S` *si*[+*se*] Assigns initial stack size (*si*) and stack expansion increment (*se*). Specify sizes in words.

`-s` Inhibits the generation of Debug Symbol tables. These are typically generated by default.

`-t` Executes in trial mode. The `segldr` utility scans all object modules and generates load maps, but it does not produce an executable program.

`-u` *unames* Enters *unames* as undefined symbols. This is useful for loading from a library, because undefined symbols are needed to force the loading of desired routines.

`-V` Lists SEGLDR's version line to the `stderr` file.

`-z` *file* Specifies an alternative default directives file. The alternative directives must configure the program correctly for execution under the UNICOS operating system.

`-Z` Inhibits the loader from reading the default directives file, either `/lib/segdirs/def_seg` or `/opt/ctl/craylibs/craylibs/segdirs/opt_defseg`. The default directives file is required for configuring programs correctly for execution under the UNICOS operating system. The `-Z` option should be used only by special-purpose programs.

files Specifies files to be loaded. These files can contain sequential object modules produced by the compilers or the assembler, or they can be object module files prepared by the `ar`(1) command or the `bld`(1) utility. Naming files on the command line has the same effect as naming them in a `BIN` directive. Files ending with `.o` will be treated as `bin` files. Files ending with `.a` will be treated as `lib` files. For compatibility, files containing directives may also be specified. The `-i` option is recommended for that purpose. It is also recommended that you create `bld` library archives rather than `ar` archives for use with Cray Research loaders.

`bin` and `lib` Files

You can direct the `segldr` utility to process an object file as either a `bin` or a `lib` file. You can specify `bin` files as arguments on the command line or by using the `BIN` directive. Name `lib` files with the `-l` option on the command line or with the `LIB` directive. The `segldr` utility processes both types of files in essentially the same manner. The `segldr` utility scans all object files and notes calling relationships. Beginning at the main program in the calling tree, `segldr` retains all modules required by the program and discards all others. Differences between `bin` and `lib` processing primarily involve the following items:

- Processing order (all `bin` files are processed before all `lib` files)
- The `-F` option or `FORCE` directive (does not affect `lib` files)
- The `-M` *file*, `brief` option, or `MAP=BRIEF` directive (lists load modules derived only from `bin` files)
- Fortran `BLOCK DATA` subprograms (always included from `bin` files but only if reference from `lib` files)
- The `DUPENTRY` directive (message level for three cases: both in `bin` files, `bin` file and `lib` file, both in `lib` files)
- The `DUPORDER` directive (if in both `bin` files and `lib` files, chooses entry point from `bin` files)
- C programs containing initialized global data (always included from `bin` files but only if referenced from `lib` files)

Both `bin` and `lib` files can contain either sequential object modules created by the compilers and the assembler, or libraries prepared by `bld`(1) and `ar`(1).

Default System Library Files

After processing all object files and any libraries supplied by the user, the `segldr` utility scans the default system library files, unless inhibited by the `-N` option or `NODEFLIB` directive. The following list shows the default library files for all Cray Research systems. The list shows the libraries in the order in which the loader searches them:

`libc.a` C library
`libu.a` Utilities library

`libm.a`	Math library
`libf.a`	Fortran library
`libfi.a`	Fortran intrinsic library
`libsci.a`	Science library
`libp.a`	Pascal library

Some of the default libraries listed may be released separately from the UNICOS operating system; therefore, they may not be present on your system. Missing libraries are silently ignored.

If you do not specify the `-L` option, the `segldr` utility looks for the default library files first in the `opt/ctl/craylibs/craylibs` and `/lib` directories, then in `/usr/lib`. If you have used one or more `-L` options, the `segldr` utility looks in all directories specified by these options, and then looks in `/lib`, `opt/ctl/craylibs/craylibs`, and `/usr/lib`. You can use the `defdir` directive to change the default directories.

The `segldr` utility usually reads directives from files supplied as option-arguments to the `-i` option. For compatibility, however, if a file provided as a command-line argument does not end in `.o` or `.a`, the `segldr` utility checks its contents. If the file contains ASCII characters, the `segldr` utility will process it as a directives file; otherwise, `segldr` processes it as an object file.

ENVIRONMENT VARIABLES

The `segldr` utility looks for and processes the following environment variables:

`SEGDIR`	Contains one or more strings separated by semicolons. Each string may be either a `segldr` directive or the name of a file containing `segldr` directives.
`TMPDIR`	Specifies the directory that the loader uses for its temporary file. The default directory may be specific to each system.
`LPP`	Specifies the number of lines to print on each page of listing output. The value must be between 15 and 999, and the default is 57.
`MSG_FORMAT`	Describes a format specification similar to that of C library routine `printf`; this specification can be used to alter `segldr` error message displays.
`NLSPATH`	Specifies a list of alternative directories that the loader should search for its error message catalog. It is used to select alternative catalogs for debugging, or when different versions of `segldr` are operating on the same system. `NLSPATH` is not needed for normal operations.
`TARGET`	Specifies the machine characteristics of the system on which the program will execute. If the `TARGET` variable has not been specified, the program will be adapted to the host system.

segldr Directives

The following is a summary of `segldr` directive syntax rules. For more complete descriptions of the directives and their use, see the *Segment Loader (SEGLDR) and ld Reference Manual*, Cray Research publication SR–0066.

- *keyword=value*
- Directives can be uppercase or lowercase but not mixed case.

- Comments can appear anywhere (an asterisk (*) indicates the beginning of a comment).
- Directives are terminated by semicolon(;), asterisk (*), or end-of-line character.
- More than one directive separated by a semicolon (;) can appear on a line.
- Directives cannot be longer than 256 characters.
- Elements in a list must be separated with commas.
- Null directives are ignored.

The following is a list of all segldr directives available under the UNICOS operating system and a brief description of each. For more complete descriptions of the directives and their use, see the *Segment Loader (SEGLDR) and ld Reference Manual*, Cray Research publication SR–0066.

abs	Specifies the file to receive the executable program.
addbss	Expands the initial size of the program.
align	Controls the starting locations of modules and common blocks.
bin	Names relocatable object input files to be searched.
calltree	Defines the start of the block of calling-tree definition directives (see endct).
callxfer	Names the external symbol used by the system startup routine to call the xfer entry.
case	Determines whether characters in the directives file are converted to uppercase before they are processed.
comment	Annotates segldr directives (* character).
commons	Loads the listed common blocks in the specified order (see scommons).
compress	Sets the threshold for compression of executable files.
copy	Forces a segmented program to execute from a scratch file.
cpucheck	Determines whether segldr performs machine-characteristic checking.
defdir	Specifies default directory search lists.
defheap	Specifies the minimum heap size and heap increment value for all programs.
deflib	Specifies extra libraries for segldr to search in addition to the default system libraries.
defstack	Sets the default program stack size.
dup	Lets segldr load modules of the same name into different segments.
dupentry	Specifies the severity level of messages for duplicated-entry point errors.
dupload	Specifies the severity level of messages for common-block initialization by more than one module.
duporder	Selects the method segldr uses to process duplicated entry points found in libraries.
dynamics	Names the common block that can expand or contract under user control.

`echo`	Resumes or suppresses the display of input directives.
`endct`	Defines the end of the block of calling-tree definition directives (see `calltree`).
`endseg`	Terminates a segment description (see `segment`).
`endtree`	Terminates the set of segment tree definition directives.
`equiv`	Substitutes a call to one entry point for a call to another.
`float`	Specifies the movable block-positioning algorithm for segmented programs.
`force`	Forces modules to be loaded, even if they are not called in execution.
`freeheap`	Specifies the minimum amount of free memory available in the heap after the initial stack allocation.
`hardref`	Converts all soft references to hard references for specified symbols.
`heap`	Allocates memory that the heap manager can manage dynamically.
`hidesym`	Specifies a global symbol that is not to be visible.
`incfile`	Identifies a symbol input file.
`include`	Identifies a directives file to be included.
`keepsym`	Specifies a global symbol that is to be visible; all other symbols are not visible.
`lbin`	Specifies relocatable object input files to be searched.
`lib`	Names the files for `segldr` to search when looking for entry points referenced in `bin` files.
`libdir`	Specifies directories other than the default to search for system libraries.
`linclude`	Specifies a file that should be included in the load process.
`llib`	Names the files for `segldr` to search when looking for entry points referenced in `bin` files using only the file name component.
`logfile`	Identifies the log messages file.
`loguse`	Identifies object files or libraries that should be logged.
`map`	Specifies the load maps to be generated by `segldr`.
`mlevel`	Specifies the severity level of messages in the listing output.
`modules`	Names the modules to be loaded (see `smodules`).
`msglevel`	Selects message severity level for specific messages.
`nodeflib`	Ignores all default libraries when loading.
`nodupmsg`	Suppresses duplicate symbol messages for specific symbols.
`nousxmsg`	Suppresses unsatisfied symbol messages for specific symbols.
`omit`	Identifies modules that should be excluded from the program.

`order` Lets you determine the central memory allocation method `segldr` uses.

`org` Sets the initial address for different portions of the program.

`outform` Specifies the type of the output file.

`preset` Specifies a value used to preset uninitialized data areas.

`redef` Specifies the severity level of messages for redefined-common-block errors.

`save` Specifies whether the current segment states for segments are written to mass storage before `segldr` overlays them with other segments.

`scanner=` `ON` | `OFF`
Scans a program targeted for a CRAY EL98 or CRAY J90 system and detects and corrects potential problems. The default on CRAY EL98 and CRAY J90 systems is `ON`.

`scanpad=` *nnnnnn*
Adds additional unused memory to a program being scanned for potential problems (see the `scanner` *directive).*

`scommons` Loads the listed common blocks in the specified order (no error messages issued).

`segment` Names the segment being described by the segment description directives (see `endseg`).

`segorder` Lets you determine the order of the segments in the executable file.

`set` Assigns a value to an entry point. You can use the `set` directive to define the library buffer size for different file structures. For more information, see the

`slt` Specifies the size of the Segment Linkage table (SLT).

`smodules` Names the modules to be loaded (no error messages issued).

`softref` Converts all hard references to soft references for specific symbols.

`stack` Sets the program stack size.

`start` Names the entry point at which the program begins executing.

`symbols` Determines whether a Debug Symbol table is generated.

`system` Selects the target operating system on which the program will be run.

`title` Specifies a page header for load maps.

`tree` Begins the set of segment tree definition directives.

`trial` Makes a sample `segldr` run without creating an executable program.

`tstack` Sets the slave task stack size for a multitasked program.

`unsat` Names unsatisfied references to be loaded from libraries.

`usx` Specifies the severity level of messages for unsatisfied-external-symbol errors.

`xfer` Names the user program entry point to which the system startup routine transfers control.

`zerocom` Specifies the name of the common block to be placed at the zero address of the data space.

`zerodata` Specifies the name of the module to be placed at the zero address of the local data space.

`zerotext` Specifies the name of the module to be placed at the zero address of the text space.

`zsyms` Specifies whether or not the loader is to include the `zzzzzz??` symbols in the load module.

MESSAGES

The full range of `segldr` error messages and the proper responses to them are listed in the *Segment Loader (SEGLDR) and ld Reference Manual*, Cray Research publication SR–0066.

FILES

`a.out`	Executable program
file`.o`	Relocatable object file
`/opt/ctl/craylibs/craylibs/libf.a`	Fortran library
`/opt/ctl/craylibs/craylibs/libfi.a`	Fortran intrinsic library
`/opt/ctl/craylibs/craylibs/libm.a`	Math library
`/opt/ctl/craylibs/craylibs/libsci.a`	Scientific library
`/lib/libc.a`	C library
`/opt/ctl/CC/CC/lib/libC.a`	C++ library (only if your site has a C++ license)
`/lib/libp.a`	Pascal library
`/opt/ctl/craylibs/craylibs/libu.a`	Utility library
`/lib/segdirs/def_seg` and `/opt/ctl/craylibs/craylibs/segdirs/opt_def_seg`	Default directives files

SEE ALSO

`ar`(1) archive and library maintainer for portable archives
`bld`(1) maintains relocatable libraries
`cc`(1) invokes the Cray Standard C compiler
`ld`(1) invokes the link editor with traditional UNIX invocation
`nm`(1) prints name list from load modules
`pascal`(1)invokes the Pascal compiler

`f90`(1) invokes the CF90 compiler

`mppld`(1) invokes the Cray Research MPP loader with traditional UNIX invocation
`mppldr`(1) invokes the Cray Research MPP loader

`a.out`(5) describes the loader output file
`mpp.a.out`(5) describes the MPP loader output file
`relo`(5) describes the relocatable object table format under the UNICOS operating system
`taskcom`(5) describes the task common table format
in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

Segment Loader (SEGLDR) and ld Reference Manual, Cray Research publication SR–0066

NAME

`setf` – Initializes a file

SYNOPSIS

`setf` [`-c`] [`-n` *size*[:*units*]] [`-p` *parts*] *file*

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `setf` utility initializes a new or existing file. The file is created if it does not already exist, and the specified number of bytes or blocks are allocated to it. By option, the user may specify that the blocks allocated must be contiguous and on which partition of the file system allocation is to be attempted.

The `setf` utility does not make changes to the inode unless preallocation is requested.

The `setf` utility accepts the following options:

`-c` Forces the program to fail if blocks cannot be allocated contiguously.

`-n` *size*[:*units*]
Indicates the total number of bytes or, if followed by the letter `b`, the total number of blocks to be allocated. The optional :*units* subfield indicates the minimum number of bytes or, if followed by the letter `b`, the minimum number of blocks to allocate per partition requested. (See the `-p` option.)

`-p` *parts* Indicates the partitions of the file system on which allocation is to be attempted. There is no guarantee that blocks will actually be allocated on the specified partitions. The *parts* field may be entered as a single number, a range (*m*–*n*), a set (*m*:*n*), or a combination of ranges and sets (see example). The dash (–) in the range specifies a range of partitions to be used (for example, 2–5 means partitions 2 through 5). A colon (:) in the set specifies a list of partitions to be used (for example, 2:4:6 means partitions 2, 4, and 6).

The partition numbers are submitted directly through `ialloc`(2) system calls. This option achieves striping of the file on the specified partitions.

files Specifies the name of the file to call or create.

NOTES

If the :*units* subfield is used on the `-n` option, and there are fewer partitions specified than will satisfy the total size if applied, (product of :*units* * *parts* is less than *size*), `setf` will repeat through the partition list in a circular fashion until the specified total size of the file is allocated. This technique is called *striding*.

If the -p option is used, and the :*units* subfield was not used on the -n option, setf will attempt to allocate the *size* equally across the total number of partitions specified by -p.

The setf command uses ialloc(2), which allocates space in multiples of allocation unit size for the file system.

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

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

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

EXAMPLES

The following setf command attempts to allocate 16,000 blocks (in units of 1000 blocks) on partitions 0 through 4, 6, and 8 through 17.

```
setf -n 16000b:1000b -p 0-4:6:8-17 myfile
```

SEE ALSO

assign(1), df(1), fck(1)

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

NAME

`setucat` – Sets your active categories

SYNOPSIS

`setucat` *cats*

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `setucat` utility activates one or more of your authorized categories. Your active category identifies the administrative role under which you are currently functioning. Your active categories are a subset of your authorized categories. Your authorized categories are initialized in the user database (UDB) by an appropriately authorized administrator.

The `setucat` utility accepts the following argument:

cats Specifies the category to be activated.

The *cats* argument consists of one of the following elements:

- Category name. The category name identifies a category to be made active.
- Comma-separated list of category names.
- A category bit mask (octal); the category bit mask is the bit value corresponding to one or more categories to be activated. This argument must be expressed as an octal number.
- The name `none`, which sets your active category to 0, is also valid. You may set your active category to 0.

The `setucat` utility can fail for one or more of the following reasons:

- The requested category is not valid.
- The requested category is not a subset of your authorized categories.

NOTES

All `setucat` requests are recorded in the security log, along with an indication of success or failure.

EXAMPLES

Example 1: In the following example, the name `syscat1` represents the category that has an octal bit mask of 040.

```
setucat 040
```

Example 2: The following examples set the `syscat1` category as your active category:

```
setucat syscat1
```

Example 3: The following example sets the `secadm` and `sysfil` categories:

```
setucat secadm,sysfil
```

Example 4: The following example deactivates an active category for a user:

```
setucat 0
```

SEE ALSO

`sh`(1), `spset`(1)

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

`slog`(4), `slrec`(5) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

General UNICOS System Administration, Cray Research publication SG–2301

NAME

setucmp – Sets your active compartments

SYNOPSIS

setucmp *cmps*

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The setucmp utility adds compartments to your active compartment set. Your active compartments determine, in part, your current file access capability. Your active compartments are a subset of your authorized compartments. Your authorized compartments are initialized in the user database (UDB) and the network access list (NAL) by an appropriately authorized administrator.

The setucmp utility accepts the following argument:

cmps Specifies compartments to be activated.

The *cmps* argument consists of one of the following elements:

- A list of one or more compartment names
- A compartment bit mask (octal)

Each compartment name identifies a compartment to be made active. Multiple compartment names must be separated by a comma (no spaces).

The compartment bit mask is the union of bit values corresponding to each compartment to be activated. This argument must be expressed as an octal number.

The *cmps* argument may also consist of the word ALL, which activates all of your authorized compartments.

The setucmp utility fails for the following error conditions:

- The requested compartments are not authorized for use on the UNICOS system.
- The requested compartments are not a subset of your authorized compartments.
- Activating the requested compartments will create an access violation with existing open files (character special files owned by the user are a special case).
- The request is not issued from the login shell process.
- There are no other processes running in the background (the process must be the master process).

NOTES

You cannot deactivate a compartment once it has been activated. In a privileged shell environment, users with an active `system` or `secadm` category are allowed to set their active compartments to any defined value. If `PRIV_SU` is enabled, the super user is allowed set its active compartments to any defined value.

The compartments of open character special files (ttys) owned by the user are automatically set to the new active compartments.

All successful requests to set your compartments are recorded in the security log and, if mandatory access violation logging is enabled, all unsuccessful requests are recorded in the security log.

EXAMPLES

In the following examples, the name `classified` represents the compartment with an octal bit mask of 020. Also, the name `confidential` represents the compartment with an octal bit mask of 04.

Example 1: The following examples add the `confidential` compartment to your active compartments:

```
setucmp confidential
setucmp 04
```

Example 2: The following examples add the `confidential` and `classified` compartments to your active compartments:

```
setucmp confidential,classified
setucmp 024
```

Example 3: The following example activates all of your authorized compartments:

```
setucmp ALL
```

SEE ALSO

`setulvl`(1), `sh`(1), `spset`(1)

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

`slog`(4), `slrec`(5) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

General UNICOS System Administration, Cray Research publication SG–2301

NAME

`setulvl` – Raises your active security level

SYNOPSIS

`setulvl` *level*

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `setulvl` utility raises your active security level. Your *active security level* is the security level at which you are currently operating, and determines, in part, your file access capability.

Your active security level is set within your authorized minimum and maximum security level range. Your authorized security level range is initialized in the user database (UDB) and the network access list (NAL) by an appropriately authorized administrator.

The `setulvl` utility accepts the following argument:

level Specifies the security level to be activated. *level* can be a number from 0 through 16, where 16 is the highest security level allowed. It can also be the name of a security level; level names are established by an appropriately authorized administrator. The requested security level must not be less than your current active security level.

The `setulvl` utility can fail for one or more of the following reasons:

- The requested level is not authorized for use on the UNICOS system.
- The requested level does not fall within the your authorized minimum and maximum security level range.
- The requested level is less than your active security level.
- The requested level creates an access violation with existing open files (character special files owned by a user are a special case).
- The request is not issued from the login shell process.
- There are other processes running in the background (for `setulvl` to execute correctly, the only process that can be running is the login shell process).

To validate your request, `setulvl` checks the minimum and maximum security levels assigned to you at login against the system's lower and upper security levels.

NOTES

You can only raise your active security level. In a privileged shell environment, users with an active `system` or `secadm` category are allowed to set their security level to any defined value. If `PRIV_SU` is enabled, the super user is allowed set its security level to any defined value.

EXAMPLES

In the following example, the name `classified` represents security level 3. The commands shown will raise your active security level to level 3:

```
setulvl classified
```

or

```
setulvl 3
```

SEE ALSO

`setucmp`(1), `sh`(1), `spset`(1)

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

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

General UNICOS System Administration, Cray Research publication SG–2301

NAME

setusrv – Sets your authorized security attributes

SYNOPSIS

setusrv [-c *valcmp*] [-i *maxcls*] [-j *valcat*] [-l *minlvl*] [-p *permit*] [-u *maxlvl*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The setusrv utility sets your authorized security attributes (that is, security levels, security compartments, integrity classes, categories, and permissions).

These authorized security attributes provide the range within which you may work. The authorized security attributes are initially determined by the security administrator.

The setusrv utility accepts the following options and arguments:

-c *valcmp* Sets your authorized compartments to *valcmp*. *valcmp* must be a comma-separated list of compartment names (no spaces) or an octal mask where each bit represents a compartment to be authorized.

-i *maxcls* Sets your maximum integrity class to *maxcls*. This option is not supported.

-j *valcat* Sets your authorized categories to *valcat*. *valcat* must be a comma-separated list of category names (no spaces) or an octal bit mask where each bit represents a category to be authorized.

-l *minlvl* Sets your minimum security level to *minlvl*. *minlvl* must be a security level number or name.

-p *permit* Sets your permissions to *permit*. *permit* must be a comma-separated list of permission names (no spaces) or an octal bit mask where each bit represents a permission to be set.

-u *maxlvl* Sets your maximum security level to *maxlvl*. *maxlvl* must be a security level number or name.

The setusrv utility can fail for one or more of the following reasons:

- An attempt is made to expand your minimum or maximum security level range.
- An attempt is made to expand your maximum integrity class.
- An attempt is made to expand your authorized compartment set.
- An attempt is made to expand your authorized category set.
- The requested maximum security level is less than the requested minimum security level.
- The requested minimum or maximum security level range is out of range of the UNICOS system minimum and maximum level range.
- The requested maximum integrity class is less than 0.

- The requested authorized compartments are out of range for the UNICOS system's set of authorized compartments.

NOTES

If the requested minimum or maximum security level or integrity class values are outside those authorized for the UNICOS system, they are silently brought within the bounds of the system.

If the requested authorized compartments, categories, or permissions are outside those authorized for the UNICOS system, they are silently brought within the bounds of the system.

All `setusrv` requests are recorded in the security log, along with an indication of success or failure.

EXAMPLES

Example 1: In the following examples, the names `classified` and `secret` represent the security levels 3 and 7, respectively. The following examples constrict your minimum or maximum security level range to levels 3 through 7:

```
setusrv -l classified -u secret
```

or

```
setusrv -l 3 -u secret
```

or

```
setusrv -l 3 -u 7
```

Example 2: In the following examples, the names `red` and `green` represent the compartments whose octal bit masks are 01 and 0400, respectively. The following examples constrict your authorized compartments to `red` and `green`:

```
setusrv -c green,red
```

or

```
setusrv -c 0401
```

Example 3: In the following examples, the names `special` and `priority` represent the categories whose octal bit masks are 020 and 0100, respectively. The following examples constrict your authorized categories to `special` or `priority`:

```
setusrv -j priority,special
```

or

```
setusrv -j 0120
```

Example 4: In the following examples, the name `suidgid` represent the permission whose octal bit masks are 0100. The following examples constrict your permissions to `suidgid`:

```
setusrv -p suidgid
```

or

```
setusrv -p 0100
```

Example 5: The following command line performs the operations illustrated by all of the preceding examples:

```
setusrv -l 3 -u 7 -c red,green -j special,priority -p 0100
```

SEE ALSO

`setucat`(1), `setucmp`(1), `setulvl`(1), `sh`(1), `spset`(1)

`setucat`(2), `setucmp`(2), `setulvl`(2), `setusrv`(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

`slog`(4), `slrec`(5) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

General UNICOS System Administration, Cray Research publication SG–2301

NAME

shrview – Displays detailed fair-share scheduler information

SYNOPSIS

shrview [-c] [-d *type*] [-o *sort*] [-r *rate*] [-s *crit*] [*ID*] [*ID*] ... [id] ...

IMPLEMENTATION

Cray PVP systems

DESCRIPTION

The shrview utility is an integrated tool for displaying information about the behavior and current state of the fair-share scheduler (also referred to as *fair-share*).

Many different display options and formats are available. Lnode information is available in several different formats, each designed to illustrate specific aspects of the fair-share scheduler. The order and selection of the lnode information can be configured. Additional displays are available to list fair-share parameters, statistics, and internal tables.

Selection and configuration of displays can be done interactively when in curses mode (see the curses(3) man page).

The shrview utility accepts the following options:

-c Enables continuous curses display.

-d *type* Selects display type. The following display types are available:

- a (ADJGROUP) Display tailored to show the effect of the ADJGROUPS flag and mingshare parameter on individual users or lnodes.
- b (brief) Abbreviated summary of user or lnode information that will fit 2 columns per 80-column page.
- c (shconsts) Dump of all fields in the kernel shconsts structure.
- l (LIMSHARE) Display tailored to show the effect of the LIMSHARE flag and maxushare parameter on individual users or lnodes.
- m (Monitor) Display of internal lnode fields related to housekeeping, which are not displayed in other displays.
- n (Nice tables) List of values in internal fair-share tables for each nice value.
- p (Params) Display of current setting of all fair-share parameters that can be set with the shradmin(8) command.
- r (Rates) Graphical representation of user or lnode information. A horizontal bar represents the relationship of current priority, user share, and current processing rate. Priority is represented with a p, user share with an s, and processing rate with a bar of # characters.

s (Statistics) Display of fair-share statistics.

v (View) Display of user or lnode information. This is the default display type. Useful indication of individual users' current fair-share priority and contributing factors.

w (Wide) A wide (132-column) display consisting of information from the view (`-dv`), `ADJGROUPS` (`-da`), monitor (`-dm`), and `LIMSHARE` (`-dl`) displays.

`-o` *sort* Selects lnode sort option. Both groups and user or account lnodes are sorted, but members of the same group will always be listed together. The following sort options are available:

c (Charge) Sorts lnodes by cumulative charges.

i (ID) Sorts lnode display based on alphabetical order of lnode names. This is the default.

p (Priority) Sorts lnodes by relative priority.

s (Share allocation) Sorts lnodes by share allocation (`rshare`).

u (Usage) Sorts lnodes by relative decayed usage.

`-r` *rate* Sets display update interval to *rate* seconds. Default is 5 seconds.

`-s` *crit* Selects the criteria for lnode or user information that is to be displayed. The following options for *crit* are available:

a (All) Display all users or lnodes. This is the default.

g (Selected groups) Display only selected groups. Enter groups to be displayed on the command line after all options.

i (Selected IDs) Display only selected IDs. Enter the selected IDs on the command line after all options.

o (Only groups) Display only resource groups.

Column Descriptions

The following columns of data are found in one or more of the `shrview` displays. Where columns with the same name have different meanings dependent on the display type, separate descriptions are provided for each display type.

Column	Description
`Adj_a`	The amount of group adjustment introduced by the `mingshare` parameter. Values of 1.0 indicated no adjustment, and values greater than 1.0 indicate increased priority for group members.
`Adj_l`	The amount of adjustment introduced by the maxushare parameter to limit the effect of past usage. Adjustments greater than 1 increase priority, and adjustments less than 1 will decrease priority.
`Chld`	Number of descendant lnodes that are members of this group.
`Chrg%`	Percentage of total cumulative charges (for selected lnodes) attributed to this ID.

CPU%	Percentage of total CPU time (for selected lnodes) attributed to this ID during the sample period.
Eshr%	Percentage of machine resources to which this user is entitled (as determined by the allocation of shares to users and groups).
Flags	An octal representation of the lnode kl.l_flags field.
Muse	The sum of the memory in use by all process that map to this lnode, expressed as clicks.
Name	Lnode user name, account, or group, as defined in the user database, with indentation to indicate hierarchy.
New%	(ADJGROUPS display only) The relative priority as adjusted by only the mingshare parameter, or actual priority without adjustments from the maxushare parameter.
New%	(LIMSHARE display only) The relative priority as adjusted by only the maxushare parameter, or actual priority without adjustments from the mingshare parameter.
Nrun	Current value of lnode kl_nrun field. This value is an estimate of the maximum amount of the rshare value (machine shares) that can be used by all runnable processes under the lnode.
Pri%	Actual priority for this ID. The priority value is relative to other selected lnodes and represents the percentage of machine resources that would be used by this ID, if all selected IDs were compute bound.
Proj%	Projected priority for this ID without the effect of the LIMSHARE or ADJGROUPS algorithms. The priority value is relative to other selected lnodes and represents the percentage of machine resources that would be used by this ID if all selected IDs were compute bound.
Rate	Current value of lnode kl_rate field. Rate is a decayed average of the number of runnable processes for this lnode.
Rate%	Percentage of total charges (for selected lnodes) ascribed to this ID during the sample period.
Ref	The number of processes mapped to this lnode, or, in the case of a group, the sum of all processes mapped to descendant lnodes.
Rshr%	Percentage of machine resources to which this user is entitled with the minimum value limited to the current value of the fair-share minimum parameter.
Svc	Service factor, an indicator of the balance between share allocation and past usage. The service factor is computed as Usg% / Shr%. A service factor greater than 1 indicates that this ID has recently received more than its allocated share and the fair-share priority will be low. A service factor less than 1 indicates that recent usage has been lower than allocated share and priority should be high.
Usage	The value of the lnode kl_usage field (represented as 1000s).
Usg%	Percentage of total decayed usage (for selected lnodes) accumulated by this ID.

Field Descriptions

The following are descriptions of fields in the statistics and nice displays.

Field	Description
`Active IDs`	The number of active IDs in the system and the maximum number of IDs
`Active groups`	The number of active groups in the system
`Usage`	The high water mark for usage values and the limit (`MAXUSAGE`)
`Share_pri`	The high water mark for `p_sharepri` values and the limit (`MAXUPRI`)
`Charge`	The percentage of total cumulative charges that are attributed to each of the cost factors
`Costs`	Current cost setting for each cost factor
`Counts`	Count of charges made for each of the cost factors
`N`	Process nice value
`Nice`	Plus or minus offset from normal nice (20)
`NiceDecays`	Priority decay rate and half-life in seconds for each nice value
`NiceRates`	A constant for each nice value at which the process rate (`kl_rate`) is incremented for each runnable process
`NiceTicks`	Rate at which a tick of CPU usage is charged for each nice value

NOTES

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

Active Category	Action
system, secadm	In a privileged administrator shell environment, allowed to write shell-redirected output to any file.
sysadm	Shell-redirected output is subject to security label restrictions.

If the PRIV_SU configuration option is enabled, the super user can write shell-redirected output to any file.

The shrview utility replaces the functionality of the shrates, shrinfo(1), shrstats(1), and shrusage(1) display commands, which are not available in the UNICOS 9.0 release.

EXAMPLES

The following example shows the use of shrview with the ADJGROUP display (-da option):

```
% shrview -da

  SHRVIEW  Type:adjgroup  Select:only groups  Sort_opt:id

     Name       Rate%   CPU%  Rshr%   Nrun  Rate  Proj%    Adj_a   New%   Pri%
------------  -----  -----  -----  -----  ----  -----  -------  -----  -----
 CCN            0.00   0.00  26.67   0.00  0.00  20.92     1.00  16.63  15.67
  SysAdm        0.00   0.00  17.78   0.00  0.00  13.82     1.00  10.99  10.35
  Syssup        0.00   0.00   8.89   0.00  0.00   7.10     1.00   5.64   5.32
 Mktg           0.31   0.23  13.33   0.00  0.00  56.65     1.00  45.05   9.99
  Country       0.09   0.00   4.44   0.00  0.00  53.91     1.00  42.87   6.66
  Intl          0.22   0.23   4.44   0.00  0.00   0.83     1.00   0.66   1.66
  TechOps       0.00   0.00   4.44   0.00  0.00   1.91     1.00   1.52   1.66
 SoftDev       97.62  99.77  60.00  65.76 21.48  22.42     1.00  38.31  74.34
  Userint      12.37  21.14  10.00  14.00 12.31   3.73     1.00   2.97   9.05
  Users         3.29   7.95  10.00  11.76  1.94  11.30     1.00  22.09  18.66
   Netdev       1.59   3.86   2.00   2.00  1.00   0.00     1.00   0.00   0.75
  Xydev        81.96  70.68  40.00  40.00  7.22   7.39     1.00  13.25  46.63
```

FILES

/usr/include/sys/share.h	Definition of shconsts structure (see share(5))
/usr/include/sys/lnode.h	Definition of lnode structure (see lnode(5))

SEE ALSO

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

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

UNICOS Resource Administration, Cray Research publication SG–2302

NAME

`sim` – Invokes an interactive Cray simulator

SYNOPSIS

`sim` [`-d`] [`-i` *dfile*] [`-m` *n*] [`-t`] [`-u`] [`-v`] [*command* [*args*]]

IMPLEMENTATION

Cray PVP systems

DESCRIPTION

The `sim` utility invokes an interactive Cray simulator. It can simulate user programs for the Cray PVP systems. The simulator builds an argument vector when the program is loaded.

The `sim` utility accepts the following options:

`-d` Starts the simulator with display of tracing off.

`-i` *dfile* Reads simulator directives from *dfile*.

`-m` *n* Sets the debug message level to *n*.

`-t` Turns on instruction timing.

`-u` Turns on vector and functional use display. Timing and trace display must also be on.

`-v` Turns off virtual memory. Usually, the simulator keeps the entire absolute file in memory but uses virtual memory pages for any BSS space and any space that the simulated program requests. This option tells the simulator to keep the entire simulated image in memory.

command Names an absolute binary file created by the loader.

args Specifies arguments for the simulated program.

Commands

Commands to `sim` are single characters followed by other optional information. Commands fall into several categories:

- Execution control (`x22`)
- Display control (`.a=100`)
- Setting memory and registers (`s0=123`)
- File manipulation (`f=file`)

The following is a summary of the available commands.

`^C` <`CONTROL-c`> interrupts simulation and prompts for more commands.

`^D` <`CONTROL-d`> causes the last command to be repeated. This is especially useful for stepping through a program several lines at a time.

`![`*command*`]` Executes shell commands. If no argument is specified, the simulator starts a shell, and suspends until the shell reads an end-of-file. If *command* is supplied, the simulator spawns a shell to execute the command.

`#[`*n*`[=`*c*`]]` Controls window displays. This command is usable only if you are using a workstation that is running the X Window System environment. Up to 10 display windows are available. The parameter *n* is a single-decimal digit indicating the window number. The parameter *c* is a single character that is the display name to use (see the following display controls). If # is entered alone, all current windows are listed. If no display indicator is specified, the window is terminated. The simulator updates all display windows whenever it displays a prompt and every 2 seconds when running.

`*[`*comment*`]` Allows commenting. This command is especially useful when you are using an alternative command file.

`?` Displays available commands and their syntaxes.

`+[`*n*`]` Scrolls the last display forward *n* octal words. The default is the display size.

`-[`*n*`]` Scrolls the last display backward *n* octal words. The default is the display size.

`.[`*a*`[=[`*add*`][`*type*`][,`*size*`][`*format*`][/`*label*`]]]`

Controls displays. There are 26 displays available. They are named `a` through `z` and can be set up individually. The displays are saved with checkpoints for convenience. A period (`.`) specified alone causes the current display setup to be printed. A period followed by a single letter (*.a*) causes that display to be printed. The `.p` display is special cased to always follow the current program address.

The *add* variable can be a global symbol, an octal address, or an A or S register indicator (for example, `!a0` uses the contents of register `A0`).

The *type* variable can be one of the following:

- `b` B register
- `c` Common memory
- `S` Shared registers
- `t` T register
- `x` Exchange package
- `v` Vector register (for example, `120v` is register `V1`, element `020`)

The *size* variable is the octal number of words to print for the display.

The *format* variable is the desired display format. The format can be one of the following:

- `b` Bit format
- `B` Byte format
- `d` Decimal format
- `f` Floating-point format
- `h` Hexadecimal format
- `i` Instruction format

p Parcel format
s Short integer format
w Word format

The optional *label* parameter allows the display to be labeled. This labeling has no effect other than to be displayed when listing displays.

/*symbol*[+*offset*][=*value*]
Displays the location of *symbol* (plus an optional *offset*) or optionally changes the contents of that location. This value may be a string of octal digits optionally followed by a parcel indicator (a, b, c, d). It may also be a quoted string (for example, 'ABC') that is entered right-adjusted, zero-filled.

<*file* Opens *file* and reads directives from it. Continues reading from stdin when an end-of-file is reached. The directive file may also specify another directive file, but there is no return to the original directive file.

>[>][:]*file* Opens trace file. This command creates the named file and begins duplicating all output onto this file. If the second > symbol is specified, all output will be appended if the file already exists. If the colon (:) is specified, instruction traces will only be written to the trace file and not to the terminal.

add[*p*]=*val* Stores a value *val* in memory address *add*. This value may be a string of octal digits optionally followed by a parcel indicator (a, b, c, d). It may also be a quoted string (for example, 'ABC') that is entered right-adjusted, zero-filled. A parcel indicator *p* may also be appended to the address to show that only one parcel is to be changed.

a*n*[=*val*] Sets or displays A register *n*. If *val* is specified, it is an octal value.

br*n*=*val* Sets B register *n*. This value may be a string of octal digits optionally followed by a parcel indicator (a, b, c, d). It may also be a quoted string (for example, 'ABC') that is entered right-adjusted, zero-filled.

b[*n*[=*padd*[/*label*][(*cond*)][;*cmds*]]]
Controls breakpoints. If b is entered alone, all current breakpoints are listed. If no parcel address is specified, the breakpoint is cleared. The parameter *n* is a single decimal digit.

The *padd* variable may be an octal parcel address (for example, "52b"), a global symbol name, or an A or S register indicator (for example, "!a0").

The *cond* variable is a condition expression. Conditional breakpoint expressions are of the form (*operand operator operand*). *operand* may be a common memory location (for example, (100) or (symbol)), an A or S register (for example, a4 or s7), or a constant. Memory addresses and constants are assumed to be octal. *operator* may be either: =, !=, <, <=, >, or >=.

The *cmds* variable is an optional string of commands, separated by semicolons, to be executed when the breakpoint is reached. These may include setting other breakpoints (this is where using a register as a breakpoint address comes in handy). This provides for a maximum of 10 active breakpoints; breakpoints remain active until cleared (breakpoint must be cleared before it is reused). The optional *label* parameter allows the breakpoint to be labeled. This labeling has no effect other than to be displayed when listing breakpoints and when the breakpoint is hit.

`c=`*file* Creates a checkpoint of the current simulator state in *file*. The file is overwritten if it already exists. All breakpoints and display setups are recorded in the checkpoint file.

`d{+,-,r}` Controls display of instruction tracing. The `+` turns on all tracing, `-` turns off all tracing, and `r` turns on tracing of only return jumps.

`f=`*file* [*args*] Loads the absolute binary *file*. This file is the output from the loader. *args* are the arguments for the simulated program.

`h` Displays a history, which is also known as a traceback.

`i<`[*file*] Redirects the simulated program's input from a file other than the terminal. If *file* is not specified, the input is read from the terminal.

`i{b,i[=0],j,o,p,t,v}`
Prints information about simulation. The following options are recognized:

- `b` Instruction buffer information (timing must be on).
- `i` Number of times each instruction was executed; the optional `=0` allows you to zero these instruction counts.
- `j` Information about conditional jump execution.
- `o` Information about various types of operations.
- `p` Virtual memory page information.
- `t` Simulator time used.
- `v` Vector memory stride statistics.

`m=`*n* Sets the debug message level to *n*. The higher the message level, the more debug messages printed.

`o>`[`>`[*file*]] Sends the standard output of the simulated program to another file. If the second `>` is specified, the data is appended. If *file* is not specified, the output is directed back to the terminal.

`p=`*padd* Sets the P register to a parcel address. *padd* may be a global symbol or an octal parcel address.

`pn=`*n* Changes the currently active processor number to processor *n*. The simulator provides the capability to simulate multitasking programs. The `_tfork`(2) system call activates another simulated processor and makes a copy of the current register and local memory contents. The simulator automatically switches between processors whenever a semaphore is cleared or when a semaphore was tested but already set. This command allows switching on demand. The register and local memory displays always display the contents of the currently active processor.

`q` Terminates the simulator (quit).

`r=`*file* Restarts the simulation from a checkpoint in *file*.

`s`*n*`=`[*val*] Sets or displays S register *n*. This value may be a string of octal digits optionally followed by a parcel indicator (`a`, `b`, `c`, `d`). It may also be a quoted string (for example, `'ABC'`) that is entered right-adjusted, zero-filled.

`sb`*n*`=`*val* Sets shared B register *n*. This value may be a string of octal digits optionally followed by a parcel indicator (`a`, `b`, `c`, `d`). It may also be a quoted string (for example, `'ABC'`) that is entered right-adjusted, zero-filled.

`st`*n*`=`*val* Sets shared T register *n*. This value may be a string of octal digits optionally followed by a parcel indicator (`a`, `b`, `c`, `d`). It may also be a quoted string (for example, `'ABC'`) that is entered right-adjusted, zero-filled.

`T`[*n*[`=`*func*]] Specifies a timing range. A timing range is an address range within which the simulator keeps track of the total time spent executing the simulated program. Instruction timing must be on for range timing to be in effect. If `T` is entered alone, all current timing ranges are listed with their current totals. If no function name is specified, the timing range is cleared. The parameter *n* is a single decimal digit.

The parameter *func* is the name of a function in the simulated program. It is used as the start address of the timing range. The simulator sets the end address of the range to the address of the next closest function. This mechanism provides for a maximum of 10 active timing ranges. Timing ranges remain active until cleared. A timing range must be cleared before it is reused.

`t{+,-}` Controls instruction timings. The `+` turns on instruction timings. It also zeroes the timer if timing was already on. The `-` turns off instruction timings.

`tr`*n*`=`*val* Sets T register *n*. This value may be a string of octal digits optionally followed by a parcel indicator (`a`, `b`, `c`, `d`). It may also be a quoted string (for example, `'ABC'`) that is entered right-adjusted, zero-filled.

`u{+,-}` Controls vector and functional unit use display. The `+` turns on the use display. The `-` turns off the use display. Instruction trace and timing must also be on. The simulator lists the vector registers and functional units that are busy along with the instruction trace. The following abbreviations are used:

0–7 = Vector registers in use

M = Floating-multiply unit in use
A = Floating-add unit in use
R = Floating-reciprocal unit in use
S = Vector-shift unit in use
I = Vector-integer unit in use
C = Common memory in use

Vn[*/format*] Displays contents of vector register *n* in format *format*.

vn.elem=val Sets V register *n*, element *elem* to a value *val*. This value may be a string of octal digits optionally followed by a parcel indicator (a, b, c, d). It may also be a quoted string (for example, 'ABC') that is entered right-adjusted, zero-filled.

vl=*val* Sets the vector length register to a decimal value *val*.

vm=*val* Sets the vector mask register to an octal value *val*.

w[*n*[=[*add1*][,*add2*][{r,w}][/*label*][(*cond*)]]]

Controls watchpoints. Watchpoints are memory addresses that are watched for a reference. If the specified address or range of addresses is referenced, the simulator stops and prints a message. If w is entered alone, all current watchpoints are listed. If no address is specified, the watchpoint is cleared. The parameter *n* is a single decimal digit.

add1 is the start address of the area to be watched. *add2* is the end address of the area to be watched. If omitted, it is the same as the start address. The optional trailing letter gives the ability to watch for only reads or only writes.

If r is specified, only memory reads are watched. If w is specified, only memory writes are watched. The default is to watch both reads and writes.

The optional *label* parameter allows the watchpoint to be labeled. This has no effect other than to be displayed when listing watchpoints and when the watchpoint is hit.

cond is a condition expression. Condition expressions are the same as for breakpoints. This mechanism provides for a maximum of 10 active watchpoints.

Watchpoints remain active until cleared. A watchpoint must be cleared before it is reused.

x[{*n*,*,e,j,R,r,s}][{+,–}]

Executes instructions. A count *n* may be specified. The default is to execute one instruction. The x command may be followed by an optional letter, which indicates a condition for stopping execution. The following options are recognized:

* Indicates infinity
e Executes to the next EXIT instruction
j Executes to the next jump instruction (this includes return jumps and EXITs)
R Executes to the return to the caller of the current routine
r Executes to the next return jump instruction
s Executes to the next process switch

The optional trailing + or - is available to override the current instruction trace status. If the - is used when instruction tracing is currently on, the simulator executes silently to the next stopping point, but tracing still remains on by default.

FILES

`/usr/bin/sim`	Cray simulator
`.simrc`	Simulator configuration file

SEE ALSO

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

NAME

`size` – Prints section sizes of executable files

SYNOPSIS

`size` [`-o`] [`-x`] *file* ...

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `size` utility produces size information in 64-bit words for each section in an absolute binary executable file. The size of the text, data, and BSS (uninitialized data) sections are printed along with their sum.

Numbers are printed in decimal, unless either the `-o` or `-x` option is used, in which case numbers are printed in octal or in hexadecimal, respectively.

The `size` utility accepts the following options and operand:

`-o` Prints numbers in octal. By default, numbers are printed in decimal.

`-x` Prints numbers in hexadecimal. By default, numbers are printed in decimal.

file ... Specifies one or more absolute binary executable files.

NOTES

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

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

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

EXAMPLES

This example lists the text, data, and BSS space of the `a.out` file:

```
$ size a.out

a.out:  63070 + 19023 + 15240 = 97333
```

SEE ALSO

`pascal`(1), `segldr`(1)

`cc`(1) in the *Cray Standard C Reference Manual*, Cray Research publication SR–2074
`f90`(1) in the *CF90 Commands and Directives Reference Manual*, Cray Research publication SR–3901

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

NAME

`sleep` – Suspends execution for a specified interval

SYNOPSIS

`sleep` *time*

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The `sleep` utility suspends execution of a process until the number of real-time seconds specified by the *time* operand have elapsed.

The `sleep` utility supports the following operand:

time A non-negative decimal integer specifying the number of seconds for which to suspend execution.

NOTES

This utility can produce error output.

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

Active Category	Action
`system`, `secadm`	In a privileged administrator shell environment, allowed to write shell-redirected output to any file.
`sysadm`	Shell-redirected output is subject to security label restrictions.

If the `PRIV_SU` configuration option is enabled, the super user can write shell-redirected output to any file.

EXIT STATUS

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

0 The execution was successfully suspended for at least *time* seconds, or a `SIGALRM` signal was received.

>0 An error occurred.

EXAMPLES

Example 1: The following example uses `sleep` to execute a command after a certain amount of time:

```
(sleep 105; command)&
```

Example 2: The following shell script fragment tests for user `joe` every 10 seconds. When `joe` logs in, the script breaks out of its loop.

```
while true
do
        echo "entering loop"
        if who | grep joe > /dev/null
        then
                break
        else
                sleep 10
        fi
done
```

SEE ALSO

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

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

NAME

sm – Invokes the UNICOS source manager (USM)

SYNOPSIS

sm [-V] [*USM_subcommand*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The UNICOS source manager (USM) is a source control system which runs under the UNICOS operating system. USM stores text and control information in USM structures called *program libraries* (PLs). When USM subcommands are run, USM uses the PLs in the current directory, unless an absolute path name is specified. For convenience, use the environment variable SMPREFIX to define the location of the PLs.

USM subcommands can be entered on the command line (*batch mode*), or USM can be invoked interactively (*interactive mode*).

In interactive mode, sm prompts for the SMPREFIX value, if undefined, and then prompts for the name of the PL on which you want to work. Thereafter, every subcommand entered will refer to this PL in the location specified by SMPREFIX. You can use the set subcommand to change either of these two values.

-V Displays the version of the sm command being used. Following is an example of the format:

```
Unicos Source Manager, Release 8.1
Change Date = 12/14/93 11:43:35
```

USM Subcommands

This subsection describes each USM subcommand and its options. The batch synopsis is given; the interactive synopsis is identical except that neither the initial sm command nor the PL is specified. For more information on USM use, see the *UNICOS Source Manager (USM) User's Guide*, Cray Research publication SG–2097.

The USM subcommands are listed alphabetically by subcommand name. Subcommand options are listed alphabetically except when they must be supplied; then the mandatory options are listed first.

sm add [-A *attribute*] [-b] [-B] [-n] [-N *named_branch*] [-o] [-r *VID*] [-z] *pl mods*
sm add -i *list* [-A *attribute*] [-b] [-B] [-n] [-N *named_branch*] [-o] [-r *VID*] [-z] *pl*

Adds a mod to the named USM PL. In interactive mode, *pl* is not specified. The add subcommand verifies that none of the files changed by the mod being applied are locked by another user through the -e option on the get subcommand. If locks are enabled, add also verifies that each file changed by the mod is locked by the current user.

If more than one mod is added with one command, the mods are treated as a single entity. Each mod must work; if not, all mods are removed.

`-i` *list* Specifies a file, *list*, that contains a list of mod names to be added to the PL. The file *list* can have a maximum of 500 entries.

`-A` *attribute* Applies the mod to the version of the file with the specified attribute.

`-b` Disables branching; specifies that a mod does not create a sideline branch.

`-B` Applies the mod to the most recent change on the sideline branch of the version specified.

`-n` Specifies retention of the edited file. (This file is usually removed when the lock is removed from the file after the mod is applied.)

`-N` *named_branch*
Applies the mod to the specified sideline branch.

`-o` Overrides lock restrictions. When locks are maintained on the PL, overrides the necessity to check out a file with intent to change before a mod can be added, or overrides a lock placed on the file by another user with the `get -e` subcommand. Only the owner of the PL can use the `-o` option.

`-r` *VID* Specifies the version ID (VID) to which the mod should be applied. This option is valid only when multiple versions of a PL are checked out.

`-z` Specifies the path name of the directory that contains the locked files changed by the added mods. If the `-z` option is not used, these files must be in the current directory.

pl (Batch mode only) The PL to which the command applies.

mods The mods to be applied to the PL. This argument is specified only when the `-i` option is not used.

`sm admin` [`-a` *login*] [`-A` *attribute*] [`-B`] [`-d` *flag* [*value*]] [`-e` *login*] [`-f` *flag* [*value*]] [`-l`] [`-L`] [`-N` *named_branch*] [`-o` *type*] [`-P`] [`-r` *VID*] [`-s`] [`-S` *VID*] [`-t`] [`-u`] [`-U`] [`-v` *VID*] [`-V`] [`-z`] *pl* [*files*]

Administers PLs. In interactive mode, *pl* is not specified. Only the owner of the PL (also referred to as the *administrator* of the PL) may use this subcommand.

`-a` *login* Adds a login name or group ID to the list of users who may make deltas (changes) to the PL. The argument *login* may be a comma-separated list of login names or group IDs. If *login* is preceded by a ! character, the specified user or group is denied permission to make changes to the PL. If the list is empty, a user with the correct access permissions to the PL is able to perform any action on the PL. In the case of ambiguities (for example, if the system has both a user ID and a group ID with the same name), USM's behavior is undefined.

`-A` *attribute* Performs the operation on the version of the module that has the specified attribute. Specifying `admin -A` *attribute* `-dA` is the the same as specifying `admin -dA`*attribute*.

`-B` Specifies that the command is to be applied to the most recent change on the sideline branch of the version specified.

`-d` *flag* [*value*]]
Disables *flag* for the PL. The following flags are available:

`A`[*attribute*] Removes the definition of the specified attribute for the modules listed. If no attribute is specified, removes all attributes from the specified version. Specifying `admin -A` *attribute* `-dA` is the the same as specifying `admin -dA`*attribute*.

`b` Allows only the PL owner (administrator) to create sideline branches.

`d` Disables the default sideline branch.

`h`[*name*] If *name* is given, removes that category from the mod header file currently in force for the PL. If *name* is omitted, disables the mod header file in force for the PL.

`i` Does not check for any string in modules in the PL.

`L` Disables hard module locks.

`m` Disables the automatic mod name generator.

`N`[*named_branch*]
Removes the definition of the specified sideline branch for the modules listed. If no sideline branch is specified, removes all sideline branches from the specified version.

`p` Disables the special substitution of the M keyword.

`s` Reverts to default mode (00444) for the source tree.

`S` Disables advisory locks.

`-e` *login* Erases a login name or group ID from the list of users who may make deltas (changes) to the PL. The argument *login* may be a comma-separated list of login names or group IDs.

`-f` *flag*[*value*]
Enables *flag* for the PL. The following flags are available:

`A`*attribute* Sets the specified attribute for the specified VID.

`b` Allows all valid PL users to create sideline branches.

`d`[*named_branch*]
Disables the sideline branch, reverting to the mainline as the default.

h[*filename*] Enables the specified mod header file; *filename* contains a description of the categories that are to go in the delta summary for the mods. If *filename* is not supplied, the `admin` command prompts interactively for the categories.

i*string* Enforces checking for *string* in each module in the PL. For example, if `-fi"%Z% %U%"` is specified, each module must contain the USM ID keywords `%Z%` (the *what line*) and `%U%` (the creation time). For more information on USM ID keywords, see the *UNICOS Source Manager (USM) User's Guide*, Cray Research publication SG–2097.

L Specifies hard module locks.

m[*no*] Sets the automatic mod name generator; *no* is optional and specifies a starting number.

N*branch* Sets the specified sideline branch for the specified VID.

p*string* Uses *string* instead of the PL name when substituting the M keyword.

s*mode* Sets the mode to be used in the source tree.

S Specifies advisory locks.

`-l` Locks the PL. This is useful when performing administrative tasks, to ensure that no user can access the PL.

`-L` Unlocks the PL, if it was locked with the `admin -l` subcommand.

`-N` *named_branch*
Performs the operation on the current version (specified by the `-A`, `-N`, or `-r` options) of the specified sideline branch.

`-o` *type* Specifies the type for updating a PL format.

`-P` Generates the specified files in the parallel source tree.

`-r` *VID* Specifies the VID of the file to which this subcommand applies.

`-s` Adds files (specified by the *files* argument) to the parallel source tree. This option is meaningful only if the PL does not maintain a full parallel source tree.

`-S` *VID* Creates a stub version for the named modules.

`-t` When regenerating the source tree, sets the time stamp for each updated file to the initial creation time of the VID. By default, the time stamp is set to the current time.

`-u` Removes files (specified by the *files* argument) from the parallel source tree. This option is meaningful only if the PL does not maintain a full parallel source tree.

`-U` Converts a PL from an old USM format to the next one. If the current PL type is "`d`", it will be updated to "`e`". If the current PL type is "`e`", it will be updated to "`f`".

`-v` *VID* Specifies the next VID for all modules to the PL. The next time a module is modified on the mainline, it will be set to this VID.

`-V` Changes all named files to the default VID. This forms a new major version; mods added before this new major version cannot be deleted. A file or a directory name must be specified with the `-V` option. If a directory name is specified, all files in that directory are included in the new major version. If a `.` symbol is specified, all files in the PL are included.

`-z` Recalculates the checksum and VIDs for the specified files.

pl (Batch mode only) The PL to which the `admin` subcommand applies.

files The individual files to which the `admin` subcommand applies. If *files* is a directory in the PL, each file in that directory is processed. If *files* is a `.` symbol, each file in the PL is processed.

`sm create` [`-l`] [`-O` *type*] [`-s`] [`-S`] [`-t`] [`-v` *VID*] *fromdir todir*

Creates a PL from *fromdir*. In interactive mode, you are prompted for the name of the PL. If *fromdir* is a PL that has UPDATE directives embedded in the code, or if it is a directory of UPDATE source files, the deck names are taken from the PL. This puts some limitations on the use of USM by users and the `-s` option of `create` must be used.

If *fromdir* is not an `UPDATE` PL, it must be a directory name. Each file in the named directory is read as input. Existing `.USM` and `.SCM` directories are ignored, which allows creation from existing USM PLs.

For a group PL, all users must have read access to the PL. This is often accomplished by creating a group for the users, then executing the `chgrp`(1) command on the PL and its contents to grant read access to the group. Another option is to create an interface, or *setuid*, program to allow access to the PL without requiring too-liberal permissions. For more information on creating an interface program, see the *UNICOS Source Manager (USM) User's Guide*, Cray Research publication SG–2097.

`-l` Creates a `locks` file. This enables the PL to lock files that are checked out with intent to change.

`-O` *type* Creates a specific format PL type. The *type* values must be "`d`", "`e`", or "`f`".

`-s` Indicates that the *fromdir* directory contains source files that have UPDATE directives already in them, and that the deck names are to be taken from the source files. If the `-s` option is used, no parallel source tree is maintained.

`-S` Enables advisory locks.

`-t` Does not create a parallel source tree.

`-v` *VID* Specifies the VID to be assigned to all files in the PL. This VID becomes the default version ID for all subsequent files added to the system.

fromdir The directory that is an `UPDATE` PL or that contains source files to be read by `create`. If *fromdir* is a `.` symbol, each file in the directory is processed.

todir The name of the new PL to be created.

`sm delete` [`-l`] [`-o`] *pl mod*

Deletes the specified mod from a PL. In interactive mode, *pl* is not specified. The `delete` subcommand is restricted to the owner of the PL.

The `delete` subcommand evaluates for dependencies all mods that are applied after the specified mod is applied. If dependencies are found, the specified mod cannot be removed. If the file is currently locked by a user, `delete` fails.

When deleting a mod on a sideline branch, the mod must be a terminal node; that is, it must be the last mod applied to that branch. A mainline mod that is the start of a sideline branch has that branch as a dependency; that mod cannot be deleted.

If the PL has been updated to a new major version number (that is, if the `admin -V` subcommand has been run on the PL) since the specified mod was added, the mod cannot be deleted without specifying the `-o` option.

`-l` Specifies that the operation to delete a mod can succeed only if the specified mod is the last mod applied to all affected modules.

`-o` Overrides lock restrictions. If locks are maintained on the PL and the mod being removed changes a file currently locked by another user, `-o` removes the lock on the file and deletes the mod anyway. Only the owner of the PL can use the `-o` option.

pl (Batch mode only) The PL to which the command applies.

mod The mod to be deleted from the PL.

`sm delta -m` *modid* [`-a`] [`-A` *attribute*] [`-b`] [`-B`] [`-e` *string*] [`-f`] [`-F`] [`-k` *string*] [`-K` *file*] [`-n`] [`-N` *named_branch*] [`-r` *VID*] [`-t` *name*] [`-u`] [`-v`] [`-y` *comment*] [`-z`] *pl files*

Makes and applies a delta (mod) to a PL. In interactive mode, *pl* is not specified.

The `delta` subcommand creates a mod to each named file, applies it to the specified PL, and returns the resulting mod to the user, in a file named *modid*. The `delta` subcommand verifies that the user who is attempting to create the delta is the same user who has the file reserved (through the `-e` option to the `get` subcommand). If the specified file does not exist, but file *file*`.u` does exist, `delta` adds the file to the PL (that is, it submits a mod to add the file to the PL).

`-m` *modid* Specifies the name of the mod to be produced. (This option must be specified unless the automatic mod name generation flag is set for the PL by using the `-fm` option of `admin`.) When `delta` completes, the mod is in a file named *modid*.

`-a` Specifies that the alternate table entries should be used when generating the mod. The table entries determine the characteristics of the mod. The alternate tables perform optimization based on different criteria from the default tables; using this option might produce a mod that is more optimal.

`-A` *attribute* Applies the mod to the version of the file that has the specified attribute.

`-b` Disables branching; specifies that the mod does not create a sideline branch.

`-B` Applies the mod to the most recent change on the sideline branch of the version specified.

`-e` *string* Specifies a string to use as the pattern when unexpanding keywords in PLs that do not support enforced keywords.

`-f` Forces the creation of a mod for all modules, whether or not they have been modified.

`-F` Allows a flexible mod header. By default, when processing information supplied with the `-k` and `-K` options, all header fields must be defined in the `USMHEADER` file. Using the `-F` option overrides this restriction.

`-k` *string* Specifies a string containing the keywords for the mod header. If the `-k` option is not used, `delta` prompts for this information. The format of the mod header is specified for the PL by using the `-fh` option of the `admin` subcommand.

`-K` *file* Specifies the file that contains the keyword information for use in the mod header.

`-n` Specifies retention of the edited file (usually removed at completion of delta processing).

`-N` *named_branch*
Applies the mod to the specified sideline branch.

`-r` *VID* Specifies the version of the PL when more than one version is checked out.

`-t` *name* Specifies a file containing the comment text to be included in the mod. If neither the `-y` or the `-t` option is used, `delta` prompts for this comment text.

`-u` Unexpands the enforced string of keywords (set with the `admin` subcommand) before making the mod.

`-v` Verbose mode. Issues messages about unchanged modules.

`-y` *comment* Specifies the comment text to include in the mod. If the `-y` option is not used, `delta` prompts for this comment text.

`-z` Uses full path names when searching for the named files. Usually, `delta` searches for the named files in the current directory.

pl (Batch mode only) The PL for which the delta is to be created.

files The files for which the mod is to be made. If *files* is a directory in the PL, each file in that directory is taken as an input file. If *files* is a `.` symbol, each file in the PL is processed. If *files* is a `-`, the list of files to process is read from standard input.

`exit`

(Interactive mode only) Exits from interactive mode.

`sm get` [`-a` *attribute*] [`-A` *attribute*] [`-b`] [`-B`] [`-c`] [`-d` *option*] [`-D`] [`-e`] [`-E`] [`-g`] [`-h`] [`-i` *VIDs*] [`-I`] [`-k`] [`-l`] [`-m`] [`-M` *mod*] [`-n`] [`-N` *named_branch*] [`-p`] [`-q`] [`-r` *VID*] [`-R` *VID*] [`-s`] [`-t`] [`-T` *date*] [`-U`] [`-v`] [`-w`] [`-x` *VIDs*] [`-z`] *pl files*

Retrieves a version of the specified files from the PL. In interactive mode, *pl* is not specified. For most options, this subcommand generates an ASCII text file from each specified file in the specified PL, according to the arguments given.

The most common option is `-e`; it returns a file that can be changed and then added to the PL as a mod with the `delta` subcommand or the `add` and `mod` subcommands. If the `-e` option is used, `get` returns two files. If the PL is set up to be pure text, *file* and *file*`.u` are returned; *file* does not need any further processing to be compiled or edited. If the PL is set up to contain `UPDATE` directives, *file*`.e` and *file*`.u` are returned; *file*`.e` must be further processed by `nupdate`(1) before it can be compiled or edited. The *file*`.u` file has a mode of 440; this is a binary control file that must not be changed or removed.

The `-n`, `-m`, `-I`, `-h`, and `-p` options provide control information in a copy of the file. For a file that supports `UPDATE` directives, there are many restrictions on these options, see the *UNICOS Source Manager (USM) User's Guide*, Cray Research publication SG–2097.

`-a` *attribute* Specifies the attribute of the version on which the delta report is based.

`-A` *attribute* Returns the version that has the specified attribute defined.

`-b` Specifies an operation on a sideline branch, if one does not yet exist. (If a sideline branch already exists, use `-r`*VID*.) This option increments the branch number of the VID.

`-B` Returns (in *file*`.d`) the most recent change on the sideline branch of the specified version.

`-c` Produces a complete report on the module when processing the `-l` and `-h` options. By default, only lines that were part of the specified VID are included in the report.

`-d` *option* Defines an option to pass on to `nupdate`(1). Use this option to define identifiers that extract a particular configuration.

`-D` Returns a delta report (in *file*`.d`) that indicates what changed after the specified version. The report indicates which lines are new and which have been deleted.

`-e` Indicates that the `get` subcommand is used to edit or make a change (mod) to the PL, followed by use of the `delta` subcommand or the `add` and `mod` subcommands. If locks are enabled, the `-e` option places a lock on the PL to prevent other `get` operations from editing the same VID. The lock remains until changes are checked in (with the `delta` or `add` subcommand), or until the the `unget` subcommand is used to return the file unchanged.

`-E` Retains the effective user ID when returning files. The default is to return files owned by the real user. (This option is useful only for effective user ID interface programs.)

`-g` Suppresses the actual retrieval of text from the USM file. It is used with the `-l` option or to verify the existence of a particular VID.

`-h` Shows each line that exists in the named files. Places the characters `<i><TAB>` at the start of any line that has been deleted from the current version of the file; each line in the retrieved file (including current and deleted lines) then has the following format:

`<i><TAB>`*line*

This information appears after the information supplied by the `-n`, `-m`, and/or `-I` options, if used (`<TAB>` is a tab character).

`-i` *VIDs* Specifies a list of VIDs or modification IDs to include in the retrieved file. This option overrides any other VID specification.

`-I` Places the modification ID at the start of each line of the retrieved files, in the following format:

modid`<TAB>`*line*

The modification ID appears after the information supplied by the `-n` and/or `-m` options, if used, and before the information supplied by the `-h` option (`<TAB>` is a tab character).

`-k` Suppresses replacement of identification (ID) keywords and processing of `UPDATE` directives in the retrieved text by their value. The `-k` option cannot be used if the `-e` option is specified.

`-l` Causes a delta summary to be written into a file named *file*`.l`. The delta summary contains mod header information and user comments added when the mod was created.

`-m` Places the VID at the start of each line of the retrieved files in the following format:

VID`<TAB>`*line*

The VID appears after the information supplied by the `-n` option, if used, and before the information supplied by the `-I` and/or `-h` options. (`<TAB>` is a tab character.)

`-M` *mod* Specifies a list of mods applied to the USM file in the PL before the file is returned to the user. This is used to test either a mod received from the field or a mod written by hand.

`-n` Places the module name at the start of each line of the retrieved files, in the following format:

module_name<TAB>*line*

The module name appears before the information supplied by the `-m`, `-I`, and/or `-h` options, if used. (<TAB> is a tab character.)

`-N` *named_branch*
Specifies that the VID of the module to be retrieved is the current VID of the specified sideline branch.

`-p` Writes the retrieved files to standard output rather than to a file. Unless the `-s` option is also used, the messages produced by `get` are redirected to standard error.

`-q` Retrieves all files changed by the mods applied with the `-M` option. This option is extremely useful for testing a list of mods that affects a large number of files.

`-r` *VID* Specifies the VID of the mod that is to be retrieved.

`-R` *VID* Specifies the version on which the delta report is based.

`-s` Suppresses the redirection of messages produced by `get`. This option is valid only when the `-p` option is also used.

`-t` Sets the time stamp for the specified file to the time the delta was created. This option is useful for PLs manipulated with makefiles; only the changed files have new dates when accessed with `get -t`.

`-T` *date* Retrieves files as they existed on the specified date.

`-U` Retrieves a `.u` file.

`-v` Verbose mode. Continues processing all modules even when an error is encountered.

`-w` Returns files with write permission enabled, even if the `-e` and `-k` options are not specified.

`-x` *VIDs* Specifies a list of VIDs or modification IDs to exclude from the retrieved file. This option overrides any other VID specification.

`-z` Puts the retrieved file in the directory (specified by *files*) rather than in the current directory.

pl (Batch mode only) The PL to which the command applies.

files The files to which the command refers in the PL. If *files* is a directory, each file in the directory is retrieved. If *files* is a `.` symbol, all files maintained in the PL are retrieved. If *files* is a `-`, the list of files is read from standard input.

`sm help` [*subcommand*]

Supplies a helpful message about the specified USM subcommand. If no subcommand is specified, `help` supplies a general help message that includes a list and a brief description of all USM subcommands.

`sm history` [`-A` *attribute*] [`-B`] [`-c`] [`-N` *named_branch*] [`-r` *VID*] [`-T` *date*] *pl files*

Writes a history of the specified files to standard output. It lists the version number, the date each version was created, the mod that creates each version, and any attributes or sideline branches assigned. If *files* is a `.` symbol, each file in the PL is processed.

`-A` *attribute* Returns history for the VID with the specified attribute.

`-B` Returns information from the sideline branch that starts at the specified VID.

`-c` Specifies that the complete history should be returned. By default, only the history of the specified VID is returned.

`-N` *named_branch*
Returns history for the VID on the specified branch.

`-r` *VID* Specifies the VID.

`-T` *date* Returns the history of the modules since the specified date.

pl (Batch mode only) The PL to which the command applies.

files The individual files to which the `history` subcommand applies. If *files* is a directory in the PL, each file in that directory is processed. If *files* is a `.` symbol, each file in the PL is processed.

`sm list` [`-A` *attribute*] [`-b`] [`-B`] [`-N` *named_branch*] [`-r` *VID*] [`-T` *date*] *pl files*

Writes a list of files in the specified PL, with the current VID of each, to standard output. In interactive mode, *pl* is not specified. If *files* is a directory, a list of all files maintained in that directory is written. If *files* is a `.` symbol, each file in the PL is listed.

`-A` *attribute* Specifies the attribute of the VID to be displayed.

`-b` Display only the module names, not the VIDs.

`-B` Returns information from the sideline branch that starts at the specified VID.

`-N` *named_branch*
Specifies the sideline branch of the VID.

`-r` *VID* Specifies the VID to be displayed.

`-T` *date* List the modules as they existed on the specified date.

pl (Batch mode only) The PL to which the command applies.

files The individual files to which the list subcommand applies. If *files* is a directory in the PL, each file in that directory is processed. If *files* is a `.` symbol, each file in the PL is processed.

`sm merge -m` *modid* [`-A` *attribute*] [`-B`] [`-e` *string*] [`-f`] [`-F`] [`-h`] [`-i`] [`-k` *string*] [`-K` *file*] [`-l`] [`-N` *named_branch*] [`-q`] [`-r` *VID*] [`-t` *name*] [`-u`] [`-y` *comment*] [`-z`] *pl files*

Merges changes for out-of-date modules when advisory locks are used.

If `merge` finds your changes can be applied, it updates the lock to the current VID and returns the merged files with names *file* and *file*`.u`. The original files are returned with names *file*`,o` and *file*`,u`. A delta report is returned in *file*`.d`. You should carefully check the suggested merged file before checking it in by using the `delta` or `mod` subcommands.

`-m` *modid* Specifies the name of the mod to be produced. (This option must be specified unless the automatic mod name generation flag is set for the PL by using the `-fm` option of `admin`.) When `merge` completes, the mod is in a file named *modid*.

`-A` *attribute* Specifies that the version to lock is the one with the specified attribute.

`-B` Specifies that the version to lock is the most recent change on the sideline branch of the version specified.

`-e` *string* Specifies a string to use as the pattern when unexpanding keywords in PLs that do not support enforced keywords.

`-f` Forces the creation of a mod for all modules, whether or not they have been modified.

`-F` Allows a flexible mod header. By default, when processing information supplied with the `-k` and `-K` options, all header fields must be defined in the `USMHEADER` file. Using the `-F` option overrides this restriction.

`-h` Specifies that the mod to be created should not have a mod header.

`-i` Information mode. Displays the names of the out-of-date modules. No mod is generated when the `-i` option is used.

`-k` *string* Specifies a string containing the keywords for use in the mod header. If the `-k` option is not used, `merge` prompts for this information. The format of the mod header to be used is specified for the PL by using the `-fh` option of the `admin` subcommand.

`-K` *file* Specifies the file that contains the keyword information for use in the mod header.

`-l` Updates the source files only; the locks file (`.USM/locks`) is not updated.

`-N` *named_branch*
Specifies that the VID of the module to be locked is the current VID of the specified sideline branch.

`-q` Specifies quick mode. This option updates only the `locks` file; it does not produce a mod and does not merge files.

`-r` *VID* Specifies the VID created by this mod.

`-t` *name* Specifies a file containing the comment text to include in the mod. If neither the `-y` nor `-t` option is used, `merge` prompts for this comment text.

`-u` Unexpands the enforced string of keywords before making the mod.

`-y` *comment* Specifies the comment text to include in the mod. If neither the `-y` nor `-t` option is used, `merge` prompts for this comment text.

`-z` Specifies that the full path name is to be used in all files specified. If the `-z` option is not used, specified files must be in the current directory.

pl (Batch mode only) The PL to which the command applies.

files The individual files to which the merge subcommand applies. If *files* is a directory in the PL, each file in that directory is processed. If *files* is a `.` symbol, each file in the PL is processed.

`sm mod -m` *modid* [`-a`] [`-A` *attribute*] [`-b`] [`-B`] [`-e` *string*] [`-f`] [`-F`] [`-h`] [`-k` *string*] [`-K` *file*] [`-N` *named_branch*] [`-r` *VID*] [`-t` *name*] [`-u`] [`-v`] [`-y` *comment*] [`-z`] *pl files*

Creates a mod from two files: *file* and *file*`.u`. In interactive mode, the PL is not specified. (If the PL supports UPDATE directives, the two files are *file*`.e` and *file*`.u`.) This mod can be applied later to the PL with the `add` subcommand.

If the specified file does not exist in the current directory, but the *file*`.u` file does exist, `mod` creates a mod to purge the specified file from the PL. If the specified file exists, and the *file*`.u` file does not exist, `mod` creates a mod to add the specified file to the PL.

`-m` *modid* Specifies the name of the mod to be created. (This option must be specified unless the automatic mod name generation flag is set for the PL by using the `-fm` option of `admin`.) When `mod` completes, the mod is in a file named *modid*.

`-a` Specifies that the alternate table entries should be used when generating the mod. The table entries dictate the characteristics of the mod. The alternate tables perform optimization based on different criteria from the default tables; using this option might produce a mod that is more optimal.

`-A` *attribute* Ensures that the VID with the specified attribute is locked when checking locks.

`-b` Prohibits `mod` operation when the version locked is a sideline branch.

`-B` Ensures that the most recent sideline change of the file is locked when checking locks.

`-e` *string* Specifies a string to use as the pattern when unexpanding keywords in PLs that do not support enforced keywords.

`-f` Forces the creation of a mod for all modules, whether or not they have been modified.

`-F` Allows a flexible mod header. By default, when processing information supplied with the `-k` and `-K` options, all header fields must be defined in the `USMHEADER` file. Using the `-F` option overrides this restriction.

`-h` Specifies that the mod to be created should not have a mod header.

`-k` *string* Specifies a string containing the keywords for the mod header. If the `-k` option is not used, `mod` prompts for this information. The format of the mod header to be used is specified for the PL by using the `-fh` option of the `admin` subcommand.

`-K` *file* Specifies the file that contains the keyword information for use in the mod header.

`-N` *named_branch*
Ensures that the current VID on the named sideline branch is locked when checking locks.

`-r` *VID* Ensures that the specified VID is locked when checking locks.

`-t` *name* Specifies a file containing descriptive text to include as the initial comment in the mod. The `-t` and `-y` options are mutually exclusive.

`-u` Unexpands the enforced string of keywords before making the mod.

`-v` Verbose mode. Issues messages about unchanged modules.

`-y` *comment* Specifies the comment text to include in the mod header. The `-y` and `-t` options are mutually exclusive.

`-z` Specifies that the full path name is to be used in all files specified. If the `-z` option is not used, specified files must be in the current directory.

pl (Batch mode only) Specifies the PL to which the mod applies.

files Specifies the files for which the mod should be generated. If *files* is a directory in the PL, each file in that directory is processed. If *files* is a `.` symbol, each file in the PL is processed. If *files* is a `-` character, the list of files to process is read from standard input.

`params`

(Interactive mode only) Displays the value of `SMPREFIX` and the name of the PL that is being modified.

`sm query` [`-A` *attribute1* [,*attribute2*]] [`-b`] [`-B`] [`-d`] [`-f`] [`-i`] [`-I`] [`-l`] [`-m`] [`-M`] [`-N` *named_branch*] [`-r` *VID*] [`-s`] [`-T` *date*] [`-u` *user*] [`-v`] *pl files*

Returns information about the current state of the specified PL.

`-A` *attribute1* [,*attribute2*]
: Specifies to show information about the version with the specified attribute or attributes. If two attributes are specified, module history information is limited to the VIDs occurring between the two attributes.

`-b`
: Displays information in an abbreviated format.

`-B`
: Specifies to show information about the version with the most recent change on the sideline branch of the version specified.

`-d`
: Displays the named branch that is currently the main line.

`-f`
: Displays a list of the flags that are enabled in the PL.

`-i`
: Lists all modules that are inactive in the list.

`-I`
: Displays the table of all known identifiers in the PL.

`-l`
: Indicates the VIDs that are locked for each module.

`-m`
: Displays a list of the mod identifiers that make up the specified VID.

`-M`
: Displays only the modules if the `-I` option is used to display a table of known identifiers.

`-N` *named_branch*
: Specifies that the VID of the module to be listed is the current VID of the specified sideline branch.

`-r` *VID*
: Specifies the VID to which the command applies.

`-s`
: Excludes modification IDs on sideline branches when displaying the identifier list.

`-T` *date*
: Returns information about the VID that existed on the specified date.

`-u` *user*
: Limits the lock information displayed to the modules locked by the specified user.

`-v`
: Displays a list of the VIDs that make up the specified VID.

pl
: (Batch mode only) The PL to which the subcommand applies.

files
: The individual files to which the subcommand applies. If *files* is a directory in the PL, each file in that directory is processed. If *files* is a `.` symbol, each file in the PL is processed.

`quit`

(Interactive mode only) Exits from `sm`.

`sm recover` *pl*

(Batch mode only) Corrects problems caused by the failure of USM subcommands. The only subcommand that is not recoverable by `recover` is the `release` subcommand. The `recover` subcommand is restricted to the owner of the PL.

`sm release` [`-p`] [`-r` *VID*] *pl* [*files*]

Removes all unused mods and inactive lines from specified files in the PL. This prepares a PL for release or distribution outside of a development group. This command is restricted to the owner of the PL.

The `release` subcommand goes through each file processed, removes support for all versions except for the most recent one, removes all unused mods from the mods directory, removes all inactive UPDATE lines from the named files, and removes the delta commentary up to this point.

`-p` Purges inactive modules from the PL history.

`-r` *VID* Removes information older than the specified version.

pl (Batch mode only) The PL to which the subcommand applies.

files The individual files to which the subcommand applies. If *files* is a directory, each file in that directory is processed. If *files* is a `.` symbol, each file in the PL is processed. If *files* is a `-` symbol, the list of files is read from standard input.

`set` [*option*] [*value*]

(Interactive mode only) Sets the value of `SMPREFIX` or the PL on which you are working.

Possible values for *option* are `SMPREFIX` and `pl`. If *option* is not specified, `sm` prompts for the value to change.

`sm unget` [`-A` *attribute*] [`-B`] [`-n`] [`-N` *named_branch*] [`-o`] [`-r` *VID*] [`-z`] *pl files*

Unlocks a version of the specified files from the PL. In interactive mode, *pl* is not specified.

`-A` *attribute* Unlocks the VID that has the specified attribute defined.

`-B` Uses the most recent VID on a sideline branch when computing VIDs.

`-n` Specifies retention of the edited file. By default, these files are removed upon successful unlocking of the module.

`-N` *named_branch*
Specifies that the VID of the module to be unlocked is the current VID of the specified sideline branch.

`-o` Forces unlocking of the version of the modules locked, regardless of the owner of the lock. Use of this option is restricted to the owner of the PL.

`-r` *VID* Specifies the VID to which the command applies.

`-s` Suppresses messages produced by this command.

`-z` Removes the unlocked files from the directory specified by the *files* argument. By default, unlocked files are removed from the current directory.

pl (Batch mode only) The PL to which the command applies.

files The files to which the command refers in the PL. If *files* is a directory, each file in the directory is retrieved. If *files* is a `.` symbol, all files maintained in the PL are retrieved. If *files* is a `-`, the list of files is read from standard input.

`sm validate` [`-e` *string*] *pl*

Checks the validity of the specified PL. In interactive mode, *pl* is not specified.

`-e` *string* Ensures that the specified string is found within all modules in the PL, in addition to the string that is enforced by the `admin` subcommand.

pl (Batch mode only) The PL to which the command applies.

!*shell_command*

(Interactive mode only) Escapes to the shell to execute a shell command.

EXAMPLES

Example 1: The following is an example of the interactive use of `sm`:

```
$ sm
sm:  SMPREFIX not set, where are your pls?
/usr/src
sm:  What is the PL you wish to work on?
prod/prog
sm:  /usr/src/prod/prog is the PL being worked on
```

Example 2: The following sequence of commands gets a list of files maintained in the PL (with `list`), extracts a file with intent to change (with `get`), changes the file (with the shell escape `!`), and then checks the changes back into the system (with `mod` and `add`):

```
SM1-> list
file1.f
dir/file2.f
file3.f
file4.f
SM2-> get -e file1.f
file1.f 1.0
next delta 1.1
SM3-> !vi file1.f          (Make changes with the editor)
SM4-> mod -m mod1 -k 'major bugfix' file1.f
comments?  (enter ^D when finished.)-----------------------
This change is an example.
^D
mod working
mod: mod1 created
SM5-> !vi mod1             (Look at mod)
SM6-> add mod1
add:  file1.f 1.1 released
changes successfully applied to prod/prog
SM7-> exit
$
```

Example 3: The following is an example of the batch (command-line) use of `sm`:

```
$ setenv SMPREFIX /usr/src
$ sm list prod/prog
file1.f
dir/file2.f
file3.f
file4.f
$ sm get -e prod/prog file1.f
file1.f 1.0
next delta 1.1
$ vi file1.f          (Make your changes)
$ sm mod -m mod1 -k 'major bugfix' prod/prog file1.f
comments?  (enter ^D when finished.)-----------------------
This change is an example.
^D
mod working
mod: mod1 created
$ vi mod1             (Look at mod)
$ sm add prod/prog mod1
add:  file1.f 1.1 released
changes successfully applied to prod/prog
$
```

SEE ALSO

`nupdate`(1)

UNICOS Source Manager (USM) User's Guide, Cray Research publication SG–2097

NAME

snmpget – Communicates with a network entity by using SNMP GET requests

SYNOPSIS

snmpget [-d] [-n *varnumber*] *host community* [-P *portnumber*] *variable-name* [*variable-name*]...

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The snmpget utility is a simple network management protocol (SNMP) application that uses the GET request to query for information on a network entity. You can specify one or more fully qualified object identifiers as arguments on the command line.

The snmpget utility accepts the following arguments:

-d	Directs the application to dump input and output packets.
-n *varnumber*	Specifies the number of variables requested per packet. This argument is useful for debugging the agent.
host	Specifies either a host name or an Internet address in dot notation.
community	Specifies the community name for the transaction with the remote system.
-P *portnumber*	Specifies an alternate port number at which to send requests to an agent.
variable-name	Specifies the fully qualified object identifier to be retrieved by the snmpget request.

EXAMPLES

The following command retrieves the sysDescr.0 and sysUpTime.0 variables:

```
snmpget  dang.cray.com  criccn  mgmt.mib.system.sysDescr.0 \
mgmt.mib.system.sysUpTime.0
```

The output is as follows:

```
Name: mgmt.mib.system.sysDescr.0
OCTET STRING- (ascii):   Kinetics FastPath2

Name: mgmt.mib.system.sysUpTime.0
Timeticks: (2270351) 6:18:23
```

If the network entity encounters an error while the request packet is being processed, an error packet is returned and a message is shown, which helps to determine the error in the request. If other variables were in the request, the request is resent without the variable in error.

SEE ALSO

RFC 1155, RFC 1157, RFC 1213

NAME

snmpgetnxt – Communicates with a network entity by using SNMP requests

SYNOPSIS

snmpgetnxt [-d] *host community* [-P *portnumber*] *variable-name* [*variable-name*]...

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The snmpgetnxt utility is a simple network management protocol (SNMP) application that uses the GET NEXT request to query for information on a network entity. You can specify one or more object identifiers as arguments on the command line. For each one, the variable that is lexicographically "next" in the remote entity's management information base (MIB) is returned.

The snmpgetnxt utility accepts the following arguments:

-d	Directs the application to dump input and output packets.
host	Specifies either a host name or an Internet address in dot notation.
community	Specifies the community name for the transaction with the remote system.
-P *portnumber*	Sends requests on port *portnumber*. You must configure the snmpd daemon to listen on this port rather than on the default port 161. This option is used for debugging.
variable-name	Specifies the fully qualified object identifier to be retrieved by the snmpgetnxt request.

EXAMPLES

The following command retrieves the sysDescr.0 and sysUpTime.0 variables:

```
snmpgetnxt  dang.cray.com  criccn  mgmt.mib.system.sysDescr.0 \
mgmt.mib.system.sysUpTime.0
```

The output is as follows:

```
Name: mgmt.mib.system.sysObjectID.0
OBJECT IDENTIFIER:   .iso.org.dod.internet.private.enterprises.cray

Name: mgmt.mib.system.sysContact.0
OCTET STRING- (ascii): John Doe doe@cray.com
```

If the network entity encounters an error while processing the request packet, an error message is shown, which helps to determine the error in the request.

SEE ALSO

RFC 1155, RFC 1157, RFC 1213

NAME

snmpnetstat – Shows network status by using SNMP

SYNOPSIS

```
snmpnetstat host community [-P portnumber]
snmpnetstat host community [-an] [-P portnumber]
snmpnetstat host community [-inrs] [-P portnumber]
snmpnetstat host community [-P portnumber] [-n] [-I interface] interval
snmpnetstat host community [-P portnumber] [-p protocol]
```

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The snmpnetstat utility symbolically displays the values of various network-related information retrieved from a remote system by using the simple network management protocol (SNMP). There are several output formats, depending on the options for the information presented. The first form of the utility displays a list of active sockets. The second form presents the values of other network-related information according to the option selected. Using the third form, with an *interval* specified, snmpnetstat continuously displays the information about packet traffic on the configured network interfaces. The fourth form displays statistics about the specified protocol.

The snmpnetstat utility accepts the following arguments:

host	Specifies either a host name or an Internet address in dot notation.
community	Specifies the community name for the transaction with the remote system.
-P *portnumber*	Sends requests on port *portnumber*. You must configure the snmpd daemon to listen on this port rather than on the default port 161. This option is used for debugging.
-a	With the default display, shows the state of all sockets; usually sockets used by server processes are not shown.
-n	Shows network addresses as numbers (usually snmpnetstat interprets addresses and attempts to display them symbolically). You can use this option with any of the display formats.
-i	Shows the state of all interfaces.
-r	Shows the routing tables. When -s is also present, shows routing statistics instead.
-s	Shows per-protocol statistics.
-I *interface*	Shows information about only the specified interface; used with the *interval* argument.
interval	Specifies interval (in seconds) through which packet traffic information is displayed.

`-p` *protocol* Shows statistics about *protocol*, which is either a well-known name for a protocol or an alias for it. Some protocol names and aliases are listed in the `/etc/protocols` file. A null response typically means that there are no interesting numbers to report. If *protocol* is unknown or if no statistics routine for it exists, the program issues a warning.

For active sockets, the default display shows the local and remote addresses, protocol, and internal state of the protocol. If a socket's address specifies a network but no specific host address, address formats are of the form `host.port` or `network.port`. When known, the host and network addresses are displayed symbolically according to the `/etc/hosts` and `/etc/networks` databases, respectively. If a symbolic name for an address is unknown, or if the `-n` option is specified, the address is printed numerically, according to the address family. For more information about the Internet dot format, see `inet`(3C). Unspecified or wildcard addresses and ports appear as `*`.

The interface display provides a table of cumulative statistics about packets transferred, errors, and collisions. The network addresses of the interface and the maximum transmission unit (MTU) are also displayed.

The routing table display indicates the available routes and their status. Each route consists of a destination host or network and a gateway to use when forwarding packets. The *flags* field shows the state of the route (`U` if up), whether the route is to a gateway (`G`), whether the route was created dynamically by a redirect (`D`), and whether the route was modified by a redirect (`M`). Direct routes are created for each interface attached to the local host; the gateway field for such entries shows the address of the outgoing interface. The interface entry indicates the network interface used for the route.

When you invoke `snmpnetstat` with an *interval* argument, it displays a running count of statistics related to network interfaces. This display consists of a column for the primary interface and a column summarizing information for all interfaces. Use the `-I` option to replace the primary interface with another interface. The first line of each screen of information contains a summary since the system was last rebooted. Subsequent lines of output show values accumulated over the preceding interval.

EXAMPLES

The following `snmpnetstat` commands produce network statistics:

```
snq1-% snmpnetstat localhost criccn -i
Name      Mtu   Network     Address            Ipkts Ierrs    Opkts Oerrs
hy0*      16432 none        none                   0     0        0     0
hy1       16432 none        none              112544     0    87800     0
vme0*     16432 none        none                   0     0        0     0
vme1*     16432 none        none                   0     0        0     0
lsx0*     16432 none        none                   0     0        0     0
hi0*      65528 none        none                   0     0        0     0
hi1*      65528 none        none                   0     0        0     0
unet0*    32880 none        none                   0     0        0     0
lo0       65535 none        none               49528     0    49534     0

snq1-% snmpnetstat localhost criccn -I hy1
Name        Mtu   Network     Address            Ipkts Ierrs    Opkts Oerrs
hy1         16432 none        none              113178     0    88523     0

snq1-% snmpnetstat localhost criccn
Active Internet Connections
Proto Recv-Q Send-Q  Local Address          Foreign Address        (state)
tcp        0      0  *.1272                *.*                    CLOSED
tcp        0      0  localhost.cray.c.1272  localhost.cray.c.sunrp TIMEWAIT
tcp        0      0  snq1.cray.com.telnet   berserkly.cray.c.1518  ESTABLISHED
tcp        0      0  snq1.cray.com.telnet   cherry28.cray.co.1934  TIMEWAIT
tcp        0      0  snq1.cray.com.telnet   fir21.cray.com.1083    ESTABLISHED
tcp        0      0  snq1.cray.com.telnet   palm15.cray.com.1093   ESTABLISHED
tcp        0      0  snq1.cray.com.telnet   palm15.cray.com.1094   ESTABLISHED
tcp        0      0  snq1.cray.com.telnet   sumac15.cray.com.1256  ESTABLISHED
tcp        0      0  snq1.cray.com.telnet   sumac15.cray.com.1257  ESTABLISHED
tcp        0      0  snq1.cray.com.telnet   sumac15.cray.com.1258  ESTABLISHED
tcp        0      0  snq1.cray.com.telnet   hose.cray.com.2946     ESTABLISHED
tcp        0      0  snq1.cray.com.login    palm03.cray.com.1021   ESTABLISHED
tcp        0      0  snq1.cray.com.login    palm10.cray.com.1022   ESTABLISHED
tcp        0      0  snq1.cray.com.login    poplar17.cray.co.1021  ESTABLISHED
tcp        0      0  snq1.cray.com.809      aspen18.cray.com.980   TIMEWAIT
tcp        0      0  snq1.cray.com.815      cherry28.cray.co.shell TIMEWAIT
```

SEE ALSO

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

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

RFC 1157

NAME

`snmpstatus` – Retrieves important information from a network entity by using SNMP requests

SYNOPSIS

`snmpstatus` [`-d`] [`-P` *portnumber*] *host community*

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `snmpstatus` utility is a simple network management protocol (SNMP) application that retrieves several important statistics from a network entity.

The `snmpstatus` utility accepts the following arguments:

`-d`	Directs the application to dump input and output packets.
`-P` *portnumber*	Sends requests on port *portnumber*. You must configure the `snmpd` daemon to listen on this port, rather than on the default port 161. This option is used for debugging.
host	Specifies either a host name or an Internet address in dot notation.
community	Specifies the community name for the transaction with the remote system. If you do not specify this argument, the community name defaults to `public`.

The information returned is as follows:

- The IP address of the entity
- A textual description of the entity (`sysDescr.0`)
- The uptime of the entity (`sysUpTime.0`)
- The sum of received packets on all interfaces (`ifInUCastPkts.* + ifInNUCastPkts.*`)
- The sum of transmitted packets on all interfaces (`ifOutUCastPkts.* + ifOutNUCastPkts.*`)
- The number of IP input packets (`ipInReceives.0`)
- The number of IP output packets (`ipOutRequests.0`)

EXAMPLES

Example 1: The following `snmpstatus` utility produces statistical information:

```
snmpstatus netdev-kbox.cc.cmu.edu public
```

The output is as follows:

```
[128.2.56.220]=>[Kinetics FastPath2] Up: 1 day, 4:43:31
IP recv/trans packets 262874/39867 |
IP recv/trans packets 31603/15805
```

Example 2: The `snmpstatus` utility also checks the operational status of all interfaces (`ifOperStatus.*`); if it finds any that are not running, it reports the interfaces as in the following example:

```
2 interfaces are down!
```

If the network entity encounters an error while processing the request packet, an error packet is returned and a message is shown, which helps to determine the error in the request. `snmpstatus` attempts to reform its request to eliminate the malformed variable, but this variable will then be missing from the displayed data.

SEE ALSO

RFC 1155, RFC 1157, RFC 1213

NAME

`snmptest` – Communicates with a network entity by using SNMP requests

SYNOPSIS

`snmptest` [`-d`] [`-P` *portnumber*] *host community*

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `snmptest` utility is a flexible simple network management protocol (SNMP) application that can monitor and manage information on a network entity.

The `snmptest` utility accepts the following arguments:

`-d`	Directs the application to dump input and output packets.
`-P` *portnumber*	Sends requests on port *portnumber*. You must configure the `snmpd` daemon to listen on this port, rather than on the default port 161. This option is used for debugging.
host	Specifies either a host name or an Internet address in dot notation.
community	Specifies the community name for the transaction with the remote system.

After invoking the program, a command-line interpreter begins to accept commands. It prompts with the following request:

```
Please enter the variable name:
```

You can enter one or more variable names, one per line. A blank line is a command to send a request for each of the variables (in one packet) to the remote entity.

EXAMPLES

In the following `snmptest` utility, the `system.sysDescr.0` name is entered at the prompt:

```
snmptest netdev-kbox.cc.cmu.edu public
Please enter the variable name:  mgmt.mib.system.sysDescr.0
Please enter the variable name:
```

The following information about the request and reply packets is returned:

```
Name: system.sysDescr.0
OCTET STRING- (ascii):
```

On startup, the program defaults to sending a GET request packet. This can be changed to a GET NEXT request or a SET request by entering the $N or $S command, respectively. Entering $G returns you to the GET request mode.

The $D command toggles the dumping of each sent and received packet.

When in SET request mode, the prompt requests more information for each variable. The following prompt requests that you enter the type of the variable:

```
Please enter variable type [i|s|x|d|n|o|t|a]:

Type "i" for an integer, "s" for an octet string in ascii, "x" for an
octet string as hex bytes separated by whitespace, "d" for an octet
string as decimal bytes separated by whitespace, "a" for an ip address
in dotted IP notation, and "o" for an object identifier.
```

You are then prompted for a value, as follows:

```
Please enter new value:
```

If it is an integer value, enter the integer (in decimal). If it is a string, enter decimal numbers separated by white space, one per byte of the string. Again, to send the packet, enter a blank line at the prompt for the variable name.

To quit the program, enter $Q at the prompt.

SEE ALSO

RFC 1155, RFC 1157, RFC 1213

NAME

snmptrap – Sends an SNMP TRAP message to a host

SYNOPSIS

snmptrap [-P *portnumber*] *host community trap-type specific-type device-description* [-a *agent-addr*] [-d]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The snmptrap utility is a simple network management protocol (SNMP) application that forms and sends an SNMP TRAP message to a host.

The snmptrap utility accepts the following arguments:

-P *portnumber* Sends requests on port *portnumber*. You must configure the snmpd daemon to listen on this port rather than on the default port 161. This option is used for debugging.

host Specifies either a host name or an Internet address in dot notation.

community Specifies the community name for the transaction with the remote system.

trap-type Specifies the type of TRAP message being sent. Trap types are integers, defined as follows:

- 0 (Cold start) The sending protocol entity is reinitializing itself such that the agent's configuration or the protocol entity implementation can be altered.
- 1 (Warm start) The sending protocol entity is reinitializing itself such that neither the agent configuration nor the protocol entity implementation is altered.
- 2 (Link down) The sending protocol entity recognizes a failure in one of the communication links represented in the agent's configuration.
- 3 (Link up) The sending protocol entity recognizes that one of the communication links represented in the agent's configuration has come up.
- 4 (Authentication failure) The sending protocol entity is the addressee of a protocol message that is not properly authenticated. While implementations of the SNMP must be able to generate this trap, they must also be able to suppress the emission of such traps through an implementation-specific mechanism.
- 6 (Enterprise specific) The sending protocol entity recognizes that some enterprise-specific event has occurred.

specific-type Identifies the particular trap that occurred.

device-description
Provides a textual description of the device sending this trap, which is used as the value of a `system.sysDescr.0` variable sent in the variable list of this trap message.

`-a` *agent-addr* Changes the address from which the trap reports it is being sent; otherwise, the sending host's address is used. This argument is optional.

`-d` Directs the application to dump the input and output packets.

EXAMPLES

The following `snmptrap` utility sends a cold start trap to the specified machine:

```
snmptrap nic.andrew.cmu.edu public 0 0
'SUN 3/60: SUNOS4.0'
```

SEE ALSO

RFC 1155, RFC 1157, RFC 1213

NAME

snmptrapd – Receives and logs SNMP TRAP messages

SYNOPSIS

snmptrapd [-p] [-d] [-P *portnumber*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The snmptrapd utility is a simple network management protocol (SNMP) application that receives and logs SNMP TRAP messages sent to the SNMP-TRAP port (162) on the local machine.

The snmptrapd utility accepts the following options:

-p Prints trap messages to the standard output; otherwise, it uses syslogd(8) to log messages. These syslog messages are sent with the level of LOG_WARNING and, if available (usually on 4.3BSD systems), they are sent to the LOG_LOCAL0 facility.

Following is an example of a log message:

```
Sep 17 22:39:52 suffern snmptrapd: 128.2.13.41: Cold Start  \
       Trap (0) Uptime: 8 days, 0:35:46
```

-d Directs the application to dump input and output packets.

-P *portnumber* Sends requests on port *portnumber*. You must configure the snmpd daemon to listen on this port rather than on the default port 161. This option is used for debugging.

The snmptrapd utility must be run as root so that UDP port 162 can be opened.

EXAMPLES

The following is an example of the use of snmptrapd. The snmpd daemon sends the coldstart trap (last line of the example) when it is started.

```
# snmptrapd -p &
# sdaemon -k snmpd
Stopping daemon: snmpd.
# sdaemon -s snmpd
Starting daemon: snmpd.
# 128.162.82.6: Cold Start Trap (0) Uptime: 0:00:00
```

SEE ALSO

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

RFC 1155, RFC 1157, RFC 1213

NAME

snmpwalk, snmpwalka – Communicates with a network entity by using SNMP requests

SYNOPSIS

snmpwalk *host community* [*variable-name*] [-d] [-P *portnumber*]

snmpwalka *host community* [*variable-name*] [-d] [-P *portnumber*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The snmpwalk command is a simple network management protocol (SNMP) application that uses GET NEXT requests to query for a tree of information about a network entity. snmpwalka performs the same function asynchronously; it does not wait for a response from the agent before issuing another request.

The snmpwalk and snmpwalka commands accept the following arguments:

host Specifies either a host name or an Internet address in dot notation.

community Specifies the community name for the transaction with the remote system.

variable-name Specifies the portion of the object identifier space that is searched, using GET NEXT requests. All variables in the subtree below the given variable are queried and their values presented to the user.

If the *variable-name* argument is not present, snmpwalk searches the whole Internet management information base (MIB).

-d Directs the application to dump input and output packets.

-P *portnumber* Sends requests on port *portnumber*. You must configure the snmpd daemon to listen on this port rather than on the default port 161. This option is used for debugging.

EXAMPLES

The following example retrieves the mgmt.mib system variables:

```
snmpwalk   netdev-kbox.cc.cmu.edu public mgmt.mib.system
```

The output is as follows:

```
Name:  system.sysDescr.0
OCTET  STRING- (ascii):   Kinetics FastPath2

Name:      system.sysObjectID.0
OBJECT      IDENTIFIER: .iso.org.dod.internet.private.enterprises.\
           CMU.sysID.CMU-KIP

Name: system.sysUpTime.0
Timeticks: (2291082) 6:21:50
```

If the network entity encounters an error while the request packet is being processed, an error packet is returned and a message is shown, which helps to determine the error in the request.

If the tree search causes attempts to search beyond the end of the MIB, the following message is displayed:

```
End of MIB.
```

SEE ALSO

RFC 1155, RFC 1157, RFC 1213

NAME

soelim – Resolves and eliminates .so requests from nroff(1) or troff(1) input

SYNOPSIS

soelim [*filename* ...]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The soelim command reads the specified files or the standard input and performs the textual inclusion implied by the nroff(1) directives of the form .so *somefile* when they appear at the beginning of input lines. This allows the placement of individual tables in separate files to be run as a part of a large document. This is useful since commands such as tbl(1) do not normally do this.

An argument consisting of a dash (-) is taken to be a file name corresponding to the standard input.

To suppress inclusion use a ´ symbol instead of a . symbol, as shown in the following example:

```
´ so /usr/ucblib/doctools/tmac/tmac.s
```

EXAMPLES

A sample usage of soelim follows:

```
soelim exum?.n | tbl | nroff -ms | col | lpr
```

SEE ALSO

nroff(1), tbl(1), troff(1)

NAME

sort – Sorts, merges, or sequence check text files

SYNOPSIS

sort [-m] [-o *output*] [-b] [-d] [-f] [-i] [-k *keydef*]... [-M] [-n] [-r] [-t *char*] [-T *dir*] [-u] [-y *kmem*] [-Y] [-z *recsz*] [*files*]

sort -c [-b] [-d] [-f] [-i] [-k *keydef*]... [-M] [-n] [-r] [-t *char*] [-T *dir*] [-u] [-y *kmem*] [-Y] [-z *recsz*] [*files*]

Obsolescent version: may not be supported in future releases:

sort [-m] [-u] [-o *output*] [-b] [-d] [-f] [-i] [-M] [-n] [-r] [-t *char*] [-T *dir*] [-y *kmem*] [-z *recsz*] [+*pos1*[-*pos2*]]... [*files*]

sort -c [-u] [-o *output*] [-b] [-d] [-f] [-i] [-M] [-n] [-r] [-t *char*] [-T *dir*] [-y *kmem*] [-z *recsz*] [+*pos1*[-*pos2*]]... [*files*]

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4
CRI extensions (-Y option)
AT&T extensions (-M, -T, and -y options)

DESCRIPTION

The sort utility performs the following functions:

- Sorts lines of all the specified files together and writes the result to the specified output.
- Merges lines of all the named (presorted) files together and writes the result to the specified output.
- Checks that a single input file is correctly presorted.

The standard input is read if you use the - as a file name or if no input files are named.

Comparisons are based on one or more sort keys extracted from each line of input. By default, there is one sort key, the entire input line, and ordering is lexicographic by bytes in machine-collating sequence.

The sort utility accepts the following options:

-c Checks that the input file is sorted according to the ordering rules; gives no output and affects only the exit code.

-m Merges only; the input files are already sorted.

-o *output* Specifies the name of an output file to use rather than the standard output. This file may be the same as one of the input files.

`-T` *dir* Specifies the name of a directory (*dir*) in which temporary files are made. If set, this overrides the `TMPDIR` environment variable. If neither the `-T` is specified nor `TMPDIR` is in the environment, temporary files are made in `P_tmpdir`, as defined in the `stdio.h` file.

`-u` Unique; suppresses all but one key in each set of lines having equal keys.

`-y` *kmem* Specifies the amount of memory (*kmem*) to be used for the sort. The amount of memory used by the sort has a large impact on its performance. Sorting a small file in a large amount of memory is a waste. If you omit this option, `sort` begins using a system default memory size, and it continues to use more as needed. *kmem* specifies the number of kilobytes of memory. If the administrative minimum or maximum is violated, the corresponding extremum will be used. Thus, `-y 0` starts with minimum memory.

`-Y` Uses the maximum amount of memory available to perform sorting.

`-z` *recsz* Records the size of the longest line read in the sort phase so that buffers can be allocated during the merge phase. If you omit the sort phase by using the `-c` or `-m` option, a popular system default size will be used. Lines longer than the buffer size causes `sort` to terminate abnormally. To prevent abnormal termination, supply the actual number of bytes in the longest line to be merged (or some larger value).

The following options override the default ordering rules. When ordering options appear independent of any key field specifications, the requested field ordering rules apply globally to all sort keys. When attached to a specific key (see `-k`), the specified ordering options override all global ordering options for that key. In the obsolescent forms, if one of more of these options follows a `+`*pos1* option, it affects only the key field specified by that preceding option.

`-d` Places in dictionary order; only letters, digits, `<space>`s, and `<tab>`s are significant in comparisons.

`-f` Considers all lowercase characters that have uppercase equivalents to be the uppercase equivalent for the purposes of comparison.

`-i` Ignores all characters that are nonprintable.

`-M` Compares as months. The first three non-`<blank>` characters of the field are folded to uppercase and compared so that `JAN` < `FEB` < ... < `DEC`. Fields that are not valid compare low to `JAN`. The `-M` option implies the `-b` option (see the following).

`-n` Restricts the sort key to an initial numeric string, consisting of optional `<blank>`s, optional minus sign, and zero or more digits with optional decimal point, which is sorted by arithmetic value. An empty digit string is treated as 0. Leading zeros and signs on zeros do not affect ordering. The `-n` option implies the `-b` option. The `-b` option is effective only when restricted sort key specifications are in effect.

`-r` Reverses the sense of comparisons.

To alter the treatment of field separators, use the following options:

`-b` Ignores leading <blank>s when determining the starting and ending positions of a restricted sort key. If you specify the `-b` option before the first `-k` option, it is applied to all `-k` options. Otherwise, the `-b` option can be attached independently to each `-k` *field_start* or *field_end* argument.

`-t` *char* Uses *char* as the field separator character; *char* is not considered to be part of a field (although it may be included in a sort key). Each occurrence of *char* is significant (for example, <*char*><*char*> delimits an empty field). If you omit the `-t` option, <blank> characters are used as default field separators; each maximal nonempty sequence of <blank> characters that follows a non-<blank> character is a field separator.

To specify sort keys, use the following options:

`-k` *keydef...* The *keydef* argument is a restricted sort key field definition. The format of this definition is as follows:

field_start[*type*][,*field_end*[*type*]]

The *field_start* and *field_end* arguments define a key field restricted to a portion of the line (see the NOTES section), and *type* is a modifier from the list of characters `b`, `d`, `f`, `i`, `M`, `n`, `r`. The `b` modifier behaves like the `-b` option, but it applies only to the *field_start* or *field_end* to which it is attached. The other modifiers behave like the corresponding options, but they apply only to the key field to which they are attached; they have this effect if specified with *field_start*, *field_end*, or both. If any modifier is attached to a *field_start* or to a *field_end*, no option applies to either. Multiple `-k` options are permitted and are significant in command line order.

When multiple key fields exist, later keys are compared only after all earlier keys compare equal. When you specify the `-u` option, lines that otherwise compare equal are ordered as if none of the options `-d`, `-f`, `-i`, `-M`, `-n`, or `-k` were present (but with `-r` still in effect, if it was specified) and with all bytes in the lines significant to the comparison.

+*pos1* (Obsolescent) Specifies the start position of a key field. (See the NOTES section.)

-*pos2* (Obsolescent) Specifies the end position of a key field. (See the NOTES section.)

NOTES

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

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

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

The following notation defines a key field that begins at *field_start* and ends at *field_end* inclusive, unless *field_start* falls beyond the end of the line or after *field_end*, in which case the key field is empty. A missing *field_end* means the last character of the line.

-k *field_start*[*type*][,*field_end*[*type*]]

A field comprises a maximal sequence of nonseparating characters and, if you omit -t, any preceding field separator.

The *field_start* portion of the *keydef* argument has the form:

field_number[.*first_character*]

Fields and characters within fields are numbered starting with 1. The *field_number* and *first_character* pieces are positive decimal integers and specify the first character to be used as part of a sort key. If .*first_character* is omitted, it refers to the first character of the field.

The *field_end* portion of the *keydef* argument has the form:

field_number[.*last_character*]

The *field_number* is as previously described for *field_start*. The *last_character* piece, interpreted as a nonnegative decimal integer, specifies the last character to be used as part of the sort key. If *last_character* evaluates to 0 or .*last_character* is omitted, it refers to the last character of the field specified by *field_number*.

If the -b option or b type modifier is in effect, characters within a field are counted from the first non-<blank> in the field. This applies separately to *first_character* and *last_character*.

The obsolescent [+*pos1* [-*pos2*]] options provide functionality equivalent to the -k *keydef* option. For comparison, the full formats of these options are as follows:

+*field0_number*[.*first0_character*][*type*] [-*field0_number*[.*first0_character*][*type*]]
-k *field_number*[.*first_character*][*type*][,*field_number*[.*last_character*][*type*]]

In the obsolescent form, fields (specified by *field0_number*) and characters within fields (specified by *first0_character*) are numbered from 0 instead of one. The optional type modifiers are the same in both forms. If .*first0_character* is omitted or *first0_character* evaluates to 0, it refers to the first character of the field. The -b option does not apply to -*pos2*.

The fully specified +*pos1* -*pos2* form with type modifiers *T* and *U*:

+*w.xT* -*y.zU*

is equivalent to the following:

undefined	(When *z* is 0, and *U* contains b and -t is present)
-k *w*+1.*x*+1*T*,*y*.0*U*	(When *z* is 0 otherwise)
-k *w*+1.*x*+1*T*,*y*+1.z*U*	(When *z* is greater than 0)

EXIT STATUS

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

0 All input files were output successfully, or `-c` was specified and the input file was sorted correctly.

1 Under the `-c` option, the file was not ordered as specified, or if the `-c` and `-u` options were both specified, two input lines that have equal keys were found. This exit status is not returned if you omit the `-c` option.

>1 An error occurred.

MESSAGES

Comments and exits with nonzero status for various trouble conditions (such as when input lines are too long), and for disorder discovered under the `-c` option.

When the last line of an input file is missing a `<newline>` character, `sort` appends one, prints a warning message, and continues.

EXAMPLES

Each of the following examples shows two command lines. The first performs the sort using the obsolescent version of `sort`; the second performs the sort using the standard `sort`.

Example 1: The following sorts the contents of `infile` with the second field as the sort key:

```
sort +1 -2 infile
sort -k 2,2 infile
```

Example 2: The following sorts, in reverse order, the contents of `infile1` and `infile2`, placing the output in `outfile` and using the first character of the second field as the sort key:

```
sort -r -o outfile +1.0 -1.2 infile1 infile2
sort -r -o outfile -k 2.1,2.2 infile1 infile2
```

Example 3: The following sorts, in reverse order, the contents of `infile1` and `infile2`, using the first non-`<blank>` character of the second field as the sort key:

```
sort -r +1.0b -1.1b infile1 infile2
sort -r -b -k 2.1,2.1 infile1 infile2
```

Example 4: The following prints the password file (`passwd`(5)), sorted by the numeric user ID (the third colon-separated field):

```
sort -t: +2n -3 /etc/passwd
sort -t: -n -k 3,3 /etc/passwd
```

Example 5: The following prints the lines of the already sorted `infile` file, suppressing all but the first occurrence of lines having the same third field (the `-um` options with just one input file makes the choice of a unique representative from a set of equal lines predictable):

```
sort -um +2 -3 infile
sort -um -k 3,3 infile
```

FILES

`/usr/tmp/stm*`	Temporary working file(s)
`/usr/tmp` `/tmp`	Default temporary directories
`TMPDIR`	User's temporary directory

SEE ALSO

`comm`(1), `csort`(1), `join`(1), `uniq`(1)

NAME

spacl – Manages an access control list (ACL)

SYNOPSIS

```
spacl -a [-i modfile] [-l] aclfile
spacl -a [-i modfile] [-s] aclfile
spacl -i modfile [-l] aclfile
spacl -i modfile [-s] aclfile
spacl -l [-t tmodfile] aclfile
spacl -r [-i modfile] [-l] aclfile
spacl -r [-i modfile] [-s] aclfile
spacl -s [-t tmodfile] aclfile
spacl -t tmodfile aclfile
```

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The spacl utility creates and maintains an access control list (ACL) file. An ACL file contains data entries that define the allowed access to a file on a specific user and/or group basis. The permissions defined in the ACL are called the absolute permissions. The absolute permissions in the ACL entries are intersected with the file's mask bits to determine the type of access allowed; this is called the effective permissions.

An ACL file consists of multiple entries, one entry per user/group name pair. The following information must be supplied for each ACL entry:

user Defines the user's login name. You can specify a valid user name, a wildcard character (*), or a question mark (?). When used in an add entry, the * represents all users. In a remove entry, the * represents only those ACL entries that specified * for the user.

The ? is used in remove entries only, to remove all ACL entries for the specified group (?:*gid*:).

group You can specify a valid group name, a wildcard character (*), a question mark (?), or a blank ().

In an add entry, the * represents all groups. In a remove entry, the * represents only those ACL entries that specified * for the group.

The ? is used in remove entries only, to remove all ACL entries for the specified user (*uid*:?:).

You can use a blank () with the wildcard character (*) to represent the owning group (*::). An entry of *:?: removes all group-only entries and the owning group entry.

permissions Permissions for access. Permissions are specified as follows:

`r` Grants read permission

`w` Grants write permission

`x` Grants execute permission

`n` Denies access

Any combination of `r`, `w`, and `x`, or `n` can be specified.

Defining duplicate entries (same *uid*`:`*gid*`:` pair) is not allowed.

There are four different types of ACL entries. Note that the format of entries are shown in pseudo code for these definitions (in the form of *user:group*). They are as follows:

User-only The absolute permissions defined for a specific user, regardless of the user's current groups. The format for this type of entry is *uid*`:*:`.

User-group The absolute permissions defined for a specific user when that user is a member of a specific group. The format for this type of entry is *uid*`:`*gid*`:`.

Group-only The absolute permissions defined for any user that is a member of the specified group. The format for this type of entry is `*:`*gid*`:`.

Owning-group The absolute permissions defined for the group that owns the file. The format for this type of entry is `*::`.

The following is a description of the file parameter types:

aclfile The *aclfile* is the file that results from the execution of `spacl` by any user. If the file does not exist, `spacl` creates it. It is stored as a binary file. The `spacl` command fails when *aclfile* is not specified or when an invalid *aclfile* is specified (that is, when the file does not conform to an `acl.h` format). Use the `spset -a` command to apply an *aclfile* to a file.

modfile The user-generated file that contains add and/or remove statements. The following is the format for adding a record via *modfile*:

`a:` *user_name* `:` *group_name* `:` *access_mode* `:`

The following is the format for removing a record via *modfile*:

`r:` *user_name* `:` *group_name* `:`

tmodfile An ASCII text that is created by using the `spget -a` or `spacl -l` command and redirecting the output to *tmodfile*. This file may be modified by using a text editor and then used as input to create the binary *aclfile*.

`spacl` accepts the following options:

`-a` *aclfile* Interactively adds entries to an ACL file named *aclfile*. You are prompted for the user's name, group name, and the access modes. Enter `quit` for user, group, or access mode to exit the interactive session. Press `<CONTROL-c>` to discard all changes and exit the interactive session.

`-i` *modfile* Inputs ACL edit statements via *modfile*.

`-l` Lists the contents of the ACL file named *aclfile*. If both the `-a` and `-l` options are specified, the listed information is the contents of *aclfile* after exiting the interactive session.

`-r` *aclfile* Interactively removes entries from an ACL file named *aclfile*. You are prompted for the user's name, group name, and the access modes. Enter `quit` for user, group, or access mode to exit the interactive session. Press `<CONTROL-c>` to discard all changes and exit the interactive session.

`-s` *aclfile* Lists the contents of *aclfile* in a short format.

`-t` *tmodfile aclfile*
Creates *aclfile* by using *tmodfile* as input.

NOTES

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

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

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

The NFS protocol was not designed to handle ACL information. The `spset`(1) command cannot set ACLs on files that reside on NFS-mounted file systems. The `spget`(1) command can display an empty ACL for all files that reside on NFS-mounted file systems.

The completion of an `spacl` command produces an updated, sorted ACL binary file named by the *aclfile* operand.

The `-a` and `-r` options cannot be processed concurrently.

EXAMPLES

Example 1: The following example shows how to interactively add entries to the ACL file called `siteacl`. The first entry adds an entry that defines absolute read and write access for `beth`, regardless of her groups. The second entry defines no access for the owning group. User input is shown in bold:

```
$ spacl -a siteacl

ENTER "quit" to END SESSION, "ctrl-c" to ABORT

enter user’s name OR an * ..... beth
enter group name OR an * ..... *
enter access mode (n = none) ...rw

ADD MODE

enter user’s name OR an * ..... *
enter group name OR a blank ...
enter access mode (n = none) ...n

ADD MODE

enter user’s name OR an * ..... quit
entry session terminated
```

Example 2: The following example shows how to interactively remove entries from an ACL called `newacl`. The first entry removes the entry that defines both the user `henry` and the group `testing`. The second entry removes all group-only entries (including the owning-group entry). User input is shown in bold:

```
$ spacl -r newacl

ENTER "quit" to END SESSION, "ctrl-c" to ABORT

REMOVE MODE

enter user's name OR a ? OR an * ...... henry
enter group name OR a ? OR an * ....... testing

REMOVE MODE

enter user's name OR a ? OR an * ...... *
enter group name OR a ? OR an * ....... ?

REMOVE MODE

enter user's name OR a ? OR an * ...... quit
entry session terminated
```

Example 3: The following example shows a file that contains edit statements for an ACL. The first line adds an entry for `bill` that allows absolute read and execute access regardless of his groups. The second line removes the entry for the user `norma` and the group `testing`. The third line adds an entry that allows no access for the owning group. The fourth line removes all user-only entries. All lines must be terminated with a colon.

```
a : bill : * : rx :
r : norma : testing :
a : * :   : n :
r : ? : * :
```

Example 4: The following example shows how to apply the file generated in the previous example (called `changes`) to the ACL file called `newacl`:

```
$ spacl -i changes newacl
```

Example 5: The following example shows the command line that applies the add/remove edit statements in `changes` to `newacl`, and then enters an interactive entry session. Upon exiting the interactive session, the contents of `newacl` is displayed:

```
$ spacl -a -l -i changes newacl
```

Example 6: An *aclfile* can be created by redirecting a *tmodfile*, as shown in the following example. The first command line shows that `tmodfile` is created from an ACL file. The second command line shows `tmodfile` being created from the ACL controls set on the file:

```
$ spacl -l newacl > tmodfile

$ spget -a foo.c > tmodfile
```

Example 7: The following example shows how to create a new ACL file called `acltest` using the `tmodfile` created in the previous example:

```
$ spacl -t tmodfile acltest
```

Example 8: The following example shows the command line that creates an ACL file named `t3` from the modfile `tlo`, then displays the contents of the new `ACL`:

```
$ spacl -l -t tlo t3
```

In the previous example, the `-s` option could have been used in place of the `-l` option, but only the short form of the contents of the ACL would be displayed.

Example 9: The following example shows the command line that creates the ACL called `myacl`, then applies the changes specified in `change` to `myacl`. The interactive add mode is entered, where you can add entries. Upon completion, the entries in `myacl` are listed:

```
$ spacl -a -l -i change myacl
```

FILES

`/usr/include/sys/acl.h`

SEE ALSO

`spclr`(1), `spset`(1)

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

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

General UNICOS System Administration, Cray Research publication SG–2301

NAME

spclr – Performs secure clear operations

SYNOPSIS

```
spclr [-m] [-i] [-r] files
spclr -d [-m] [-i] [-r] [-p] files
spclr -s [-m] [-i] [-r] files
spclr -a [-m] [-i] [-r] [-F] [-V] files
spclr -M [-K] [-a] [-m] [-i] segments
spclr -Q [-K] [-a] [-m] [-i] queues
spclr -S [-K] [-a] [-m] [-i] semaphores
```

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The spclr utility removes files and overwrites (clears) associated disk space with a pattern. The spclr utility performs disk space clearing and declassification procedures in accordance with Department of Defense guidelines. This utility also removes access control lists (ACLs) assigned to files, directories, or IPC objects (shared memory segments, message queues, and semaphores).

If spclr is called without the -d, -s or -a option, it functions similar to the rm(1) utility. In other words, the file is removed, but the disk space associated with the file is not overwritten.

The -F and -V options are available only if the Cray/REELlibrarian (CRL) has been installed on the UNICOS system.

The spclr utility accepts the following options:

-F Interprets *files* as a list of CRL files. The format of the elements of *files* is one of the following:

```
V:volume_set_name  file_name
V:volume_set_name S:file_sequence
file_name
```

The *volume_set_name* variable is an existing CRL volume set name or *.vsid*, where *vsid* is the first volume ID of a volume set. *file_sequence* is the numeric file sequence of the CRL file. *file_name* is the file ID of the CRL file. The V:*volume_set_name S*:*file_sequence* format is always unambiguous, whereas the other two formats are not.

-V Interprets *files* as a list of CRL volume sets. The format of the elements of *files* is one of the following:

```
volume_set_name
.vsid
```

The *volume_set_name* variable is an existing CRL volume set name. *vsid* is the first volume ID of a volume set.

`-M` *segments*
Interprets *segments* as a list of shared memory segments. By default, these are named using the shared memory identifiers displayed by the `ipcs`(1) command. See the `-K` option explanation.

`-Q` *queues*
Interprets *queues* as a list of message queues. By default, these are named using the message queue Identifiers displayed by the `ipcs`(1) command. See the `-K` option explanation.

`-S` *semaphores*
Interprets *semaphores* as a list of semaphores. By default, these are named using the semaphore identifiers displayed by the `ipcs`(1) command. See the `-K` option explanation.

`-K` Specifies that the message queue, shared memory, or semaphore names specified in the arguments are keys as displayed by `ipcs`(1), not identifiers.

`-d` Declassifies disk space associated with *files*. Space is overwritten with the `DECLASSIFY_PATTERN` configuration parameter, then with the negated pattern, then with the original pattern. This option is available only when the `DECLASSIFY_DISK` configuration parameter is enabled. It cannot be used with the `-a` or `-s` options.

`-p` Uses a random pattern when declassifying disk space associated with *files*. Space is overwritten with the `DECLASSIFY_PATTERN` configuration parameter, then with the negated pattern, then with a random pattern. This option is valid only with the `-d` option.

`-m` Disables printing of error messages.

`-i` Requests permission before removing a file or the file's ACL.

`-r` Recursively removes the contents of a directory, its subdirectories, and the directory itself. If the `-a` option is specified, only the ACLs of the contents of a directory, its subdirectories, and the directory itself are recursively removed.

`-s` Sanitizes disk space associated with *files*. Space is filled with a pattern that is determined by the `SANITIZE_PATTERN` configuration parameter. Cannot be used with `-d`, `-a`, or `-p` options.

`-a` Removes ACL associated with *files*. Cannot be used with `-d`, `-s`, or `-p` options.

The group access mode is set to the mode granted the owning group before the ACL is removed (that is, the mode granted when the ACL was set on the object).

The `setuid` and `setgid` bits are cleared if the caller is not appropriately authorized.

NOTES

When the path name supplied to the `spclr` utility specifies a multilevel symbolic link (the name of a multilevel directory), the attributes are changed only on the root of the multilevel directory tree. In the case of ACLs, this affects all subsequently created labeled subdirectories. In any case, the attribute change does not affect existing labeled subdirectories. To set attributes on the existing labeled subdirectories, you must specify the path names of the existing labeled subdirectory found in the root of the multilevel directory to the `spclr` utility.

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

Active Category	Action
`system`, `secadm`	Allowed to perform all `spclr` operations on any file. For the `-a` option, the file `setuid` and `setgid` bits are not cleared.
`sysadm`	Allowed to perform all `spclr` operations on files, subject to security label restrictions. For the `-a` option, the file `setuid` and `setgid` bits are not cleared.

If `PRIV_SU` is enabled, the super user is allowed to perform all `spclr` operations on any file. The super user or a user with the `suidgid` permission can override the clearing of the file's `setuid` and `setgid` bits.

An appropriately authorized administrator can set the declassify pattern and number of overwrites during UNICOS system configuration. The administrator can also change the sanitize pattern at this time. See *General UNICOS System Administration*, Cray Research publication SG–2301.

SEE ALSO

`ipcs`(1), `rm`(1), `spacl`(1), `spset`(1)

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

Cray/REELlibrarian (CRL) User's Guide, Cray Research publication SG–2126

Cray/REELlibrarian (CRL) Administrator's Guide, Cray Research publication SG–2127

General UNICOS System Administration, Cray Research publication SG–2301

Department of Defense Magnetic Remanence Security Guideline, CSC-STD-005-85, November 15, 1985

NAME

split – Splits files into pieces

SYNOPSIS

split [-l *line_count*] [-a *suffix_length*] [-v] [*file* [*name*]]
split -b *n*[*unit*] [-a *suffix_length*] [-v] [*file* [*name*]]

Obsolescent version: May not be supported in future releases:

split [-*line_count*] [-a *suffix_length*] [-v] [*file* [*name*]]

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The split utility reads *file* and writes it in *n*-line pieces (default 1000 lines) onto a set of output files (*name*). To modify the size of the output files, use the -b or -l option. Each output file is created with a unique suffix, which consists of exactly *suffix_length* lowercase letters from the POSIX local(2). By default, the names of the output files are x, followed by a two-character suffix, starting with aa, ab, ac, and so on, and continuing until the suffix zz, for a maximum of 676 files.

If the number of files required exceeds the maximum allowed by the suffix length provided, split fails after creating the last file with a valid suffix.

The split utility accepts the following options and operands:

-a *suffix_length*
: Use *suffix_length* letters to form the suffix part of the file names of the split file. If you omit -a, the default suffix length is 2. If the sum of the *name* operand and the *suffix_length* argument would create a file name that exceeds {NAME_MAX} bytes, an error occurs; split then exits and no files are created.

-b *n*[*unit*]
: Split a file into pieces *n* bytes in size. If a *unit* is k, the file is split into pieces *n**1024 bytes in size. If a *unit* is m, the file is split into pieces *n**1048576 bytes in size.

-l *line_count*
-*line_count* (Obsolescent)
: Indicates the number of lines in each piece. Default is 1000 lines. If the input does not end with a <newline>, the partial line is included in the last output file.

-v
: Writes out each output file name to standard output as it is created.

file Specifies the path name of the ordinary file to be split. If you do not specify an input file or *file* is `-`, the standard input is used.

name Specifies the prefix to be used for each of the files resulting from the split operation. If you omit the *name* argument, `x` is used as the prefix of the output files. The combined length of the basename of *prefix* and *suffix_length* cannot exceed {`NAME_MAX`} bytes.

NOTES

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

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

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

EXIT STATUS

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

0 Successful completion.

>0 An error occurred.

SEE ALSO

`csplit`(1)

NAME

spset, spget – Sets and displays security attributes

SYNOPSIS

```
spset [-a aclfile] [-c cmp] [-d facl] [-f] [-i cls] [-j cat] [-k flgs] [-l lvl] [-F] [-V] files
spset -M [-K] [-a aclfile] [-c cmp] [-d facl] [-f] [-l lvl] segments
spset -Q [-K] [-a aclfile] [-c cmp] [-d facl] [-f] [-l lvl] queues
spset -S [-K] [-a aclfile] [-c cmp] [-d facl] [-f] [-l lvl] semaphores
spset -s min max cmps

spget [-a[r][-e]] [-f] [-F] [-V] files
spget -M [-K] [-a[r][-e]] [-f] segments
spget -Q [-K] [-a[r][-e]] [-f] queues
spget -S [-K] [-a[r][-e]] [-f] semaphores
spget [-s]
```

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The spset and spget utilities allow you to manipulate the security environment. Use spset to set attributes and spget to display those attributes. Attempts by unauthorized users to change security attributes can be audited.

The -F and -V options are available only if the Cray/REELlibrarian (CRL) has been installed on the UNICOS system.

The spset utility accepts the following options:

-a *aclfile* Assigns the access control list (ACL) to the indicated files and/or directories. *aclfile* is a file containing the ACL (created with spacl(1)) and *files* is one or more file and/or directory names to which the ACL is assigned. The -a option can fail for the following reasons:

- You are not the owner of the *files* or you are not appropriately privileged.
- Either the *files* or *aclfile* do not exist (are not valid).
- The *aclfile* file is not a legitimate ACL.

The group access mode is set to the mode granted the owning group before the ACL is removed (that is, the mode granted when the ACL was set on the object).

The setuid and setgid bits are cleared if the caller is not appropriately privileged.

Note that the `spset -a` utility does not result in any connection between the *aclfile* and the *files*. The data from the *aclfile* is converted to binary form and moved to a block named in the inode. Thus, if *aclfile* is modified or removed after `spset -a` is executed, the ACL permissions assigned to *files* are not changed. To change the ACL permissions, another ACL must be assigned to the file. To remove an ACL from a file, the `spclr`(1) utility must be used.

`-c` *cmp* Sets the file compartments (*cmp*) for one or more files. The requester must be an appropriately authorized user. *cmp* can be either the octal representation of specific compartments or a string of one or more (comma-separated) compartment names as defined in `uts/cf/seclabs.c`. The user must specify compartments within the authorized set for the file system on which the file resides.

If the `-V` option is used, *cmp* can be in the *lower_set upper_set* format if you are setting a multilevel CRL volume set. *upper_set* must dominate *lower_set*.

`-d` *facl* Assigns an ACL to the indicated files and/or directories by duplicating the ACL permissions assigned to *facl*. All rules pertaining to the `-a` option apply. If the `-M`, `-Q`, or `-S` option is specified, the *facl* argument is interpreted as a file name and not as an IPC object name.

`-f` Forces the assignment of a new ACL. If a file already has an ACL, `spset` asks whether the existing ACL should be replaced if the `-f` option is not selected. Using the `-f` option suppresses the question and replaces the ACL.

`-i` *cls* Sets the file integrity class for one or more files. The requester must be an an appropriately authorized user. This option is not supported.

`-j` *cat* Sets the file categories for one or more files. The requester must be an appropriately authorized user. This option is not supported.

`-k` *flgs* Sets the file security flags for one or more files. The requester must be an an appropriately authorized user. *flgs* can be either the octal representation of specific security flags or a string of one or more (comma-separated) flag names as defined in `/usr/include/sys/tfm.h`. The `exec` flag is obsolete. It is not possible to enable the secure device flag (`secdv`) using `spset`. To enable this flag, the `spdev`(8) command must be used.

`-l` *lvl* Sets the file security level (*lvl*) for one or more files. The requester must be an appropriately authorized user. The file's security level must fit within the boundaries established by the lower and upper security levels of the file system in which the file resides. *lvl* can be either a decimal integer or a mnemonic level name, as defined in `uts/cf/seclabs.c`.

If the `-V` option is used, and you are setting a multilevel CRL volume set, the format can be *lower-upper*.

`-M` *segments* Interprets *segments* as a list of shared memory segments. By default, these are named using the shared memory identifiers displayed by the `ipcs`(1) command. See the `-K` option explanation.

`-Q` *queues* Interprets *queues* as a list of message queues. By default, these are named using the message queue identifiers displayed by the `ipcs`(1) command. See the `-K` option explanation.

-S *semaphores*
Interprets *semaphores* as a list of semaphores. By default, these are named using the semaphore identifiers displayed by the ipcs(1) command. See the -K option explanation.

-K Specifies that the message queue, shared memory, or semaphore names specified in the arguments are keys as displayed by ipcs(1), not identifiers.

-F Interprets *files* as a list of CRL files. The format of the elements of *files* is one of the following:

V:*volume_set_name file_name*
V:*volume_set_name S*:*file_sequence*
file_name

The *volume_set_name* variable is an existing CRL volume set name or *.vsid*, where *vsid* is the first volume ID of a volume set. *file_sequence* is the numeric file sequence of the CRL file. *file_name* is the file ID of the CRL file. The V:*volume_set_name S*:*file_sequence* format is always unambiguous, whereas the other two format are not.

-V Interprets *files* as a list of CRL volume sets. The format of the elements of *files* is one of the following:

volume_set_name
.*vsid*

The *volume_set_name* variable is an existing CRL volume set name. *vsid* is the first volume ID of a volume.

files Specifies the file that contains the attributes that are being changed.

-s *min max cmps*
Sets the minimum and maximum system security level and the system's authorized compartments. The requester must be an an appropriately authorized user. *cmps* can be either the octal representation of specific compartments or a string of one or more (comma-separated) compartment names as defined in uts/cf/seclabs.c. In addition, the following rules must be satisfied:

- *min* must be less than or equal to the minimum security level for all mounted file systems, with the exception of file systems that are labeled with the syslow security label.
- *max* must be greater than or equal to the maximum security level for all mounted file systems, with the exception of file systems that are labeled with the syshigh security label.
- *cmps* must dominate all authorized compartments for all mounted file systems.

The spget utility accepts the following options:

No options Displays the user's security environment (permissions, security levels, compartments, integrity class, and categories). The active and maximum integrity class fields are not supported.

`-a` Displays the ACL information for the specified files.

`-r` Displays the ACL information in reduced format. Valid only with the `-a` option.

`-f` Displays the security attributes (security level, compartments, integrity class, categories, and flags) for the specified objects. If you specify both the `-a` and `-f` options, both the ACL and the security attribute information is displayed. Message queues, shared memory segments, and semaphores have only security levels and compartments, so only these security attributes are displayed. If you specify the `-M`, `-S`, or `-Q` option, but do not specify the `-a` option, specifying the `-f` option is not needed. The file category and class fields are not supported.

`-s` Displays the system security minimum and maximum security levels and authorized compartments.

`-e` Displays the masked ACL permissions. Valid only with the `-a` option. This option masks each ACL entry's mode against the file's permissions and displays the resultant mode.

`-M` *segments* Obtains the attributes of the shared memory segments named in *segments*. By default, these are named using the shared memory identifiers displayed by the `ipcs`(1) command. See the `-K` option explanation.

`-Q` *queues* Obtains the attributes of the message queues named in *queues*. By default, these are named using the message queue identifiers displayed by the `ipcs`(1) command. See the `-K` option explanation.

`-S` *semaphores*
Obtains the attributes of the semaphores named in *semaphores*. By default, these are named using the semaphore identifiers displayed by the `ipcs`(1) command. See the `-K` option explanation.

`-K` Specifies that the message queue, shared memory, or semaphore names specified in the arguments are keys as displayed by `ipcs`(1), not identifiers.

`-F` Interprets *files* as a list of CRL files. See the description in the `spset` options list.

`-V` Interprets *files* as a list of CRL volume sets. See the description in the `spset` options list.

Compartments, permissions, and categories are displayed in octal and by name. Levels and classes are displayed in decimal and by name. Your active security attributes are those at which you are currently operating.

NOTES

When the path name supplied to the `spset` utility specifies a multilevel symbolic link (the name of a multilevel directory), the attributes are changed only on the root of the multilevel directory tree. In the case of ACLs, this affects all subsequently created labeled subdirectories. In any case, the attribute change does not affect existing labeled subdirectories. To set attributes on the existing labeled subdirectories, you must specify the path names of the existing labeled subdirectory found in the root of the multilevel directory to the `spset` utility.

If the level and compartments are both to be set, the spdev(8) command must be used, since spset does not set level and compartments at the same time. Once the level or compartment has changed, the user loses write access to the directory and the other attributes can no longer be set.

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

Active Category	Action
system, secadm	Allowed to perform all spset and spget operations on any file.
sysadm	Allowed to set and display file access control lists and display file security attributes, subject to security label restrictions. Allowed to retrieve system security attributes.

If the PRIV_SU configuration option is enabled, the super user is allowed to perform all spset and spget operations on any file.

EXIT STATUS

Returns 0 on successful completion; otherwise, returns nonzero.

EXAMPLES

Example 1: The following example sets the security level of the shared memory segments whose IPC keys are 0x12345 and 0x6789a to 5:

```
spset -l 5 -K -M 0x12345 0x6789a
```

Example 2: The following example displays the security labeling information on the shared memory segments whose IPC keys are 0x12345 and 0x6789a:

```
spget -K -M 0x12345 0x6789a
```

Example 3: The following example displays the ACL on the shared memory segments whose IPC keys are 0x12345 and 0x6789a:

```
spget -K -M -a 0x12345 0x6789a
```

Example 4: The following example sets the security level of the shared memory segments whose IPC identifiers are 1234 and 5678 to 5:

```
spset -l 5 -M 1234 5678
```

Example 5: The following example displays the security labeling information on the shared memory segments whose IPC identifiers are 1234 and 5678:

```
spget -M 1234 5678
```

Example 6: The following example displays the ACL on the shared memory segments whose IPC identifiers are `1234` and `5678`:

```
spget -M -a 1234 5678
```

Example 7: The following example displays the ACL and security labeling information on the shared memory segments whose IPC identifiers are `1234` and `5678`:

```
spget -M -a -f 1234 5678
```

Example 8: The following example displays both the ACL and the security labeling information on the shared memory segments whose IPC keys are `0x12345` and `0x6789a`:

```
spget -K -M -a -f 0x12345 0x6789a
```

You can substitute the `-S` or `-Q` options for the `-M` option in the previous examples to change from shared memory segments to semaphores or message queues information respectively, being set or displayed.

The previous examples use hexadecimal values for the keys and decimal values for the identifiers, because this is how they are typically displayed by the `ipcs`(1) command. Any decimal, octal, or hexadecimal value can be used for either the key or the identifier, as long it is specified in the standard form: 0xxxxx or 0Xxxxx for hexadecimal, 0xxxx for octal, and [1–9]xxxx for decimal.

SEE ALSO

`ipcs`(1), `setucmp`(1), `setulvl`(1), `spacl`(1), `spclr`(1)

`getfacl`(2), `getsysv`(2), `getusrv`(2), `secstat`(2), `setfacl`(2), `setfcmp`(2), `setfflg`(2), `setflvl`(2), `setsysv`(2), `setucat`(2), `setucmp`(2), `setulvl`(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

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

Cray/REELlibrarian (CRL) User's Guide, Cray Research publication SG–2126

Cray/REELlibrarian (CRL) Administrator's Guide, Cray Research publication SG–2127

General UNICOS System Administration, Cray Research publication SG–2301

NAME

strings – Finds printable strings in files

SYNOPSIS

strings [-a] [-f] [-n *number*] [-t *format*] [*files*]

Obsolescent version; may not be supported in future releases;

strings [-] [-f] [-o] [-t *format*] [-*number*] [*files*]

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4
BSD extensions (-f and -o options)

DESCRIPTION

The strings utility looks for printable strings in regular *files* and writes those strings to standard output. A character string is any sequence of 4 (the default) or more printable characters that end with a <newline> or a NULL character.

The strings utility accepts the following options:

-a
- (Obsolescent) Scans files in their entirety. If -a is not specified, the executable file starts after the scan of an exec(2) structure.

-f Precedes each string by the name of the file in which it was found.

-n *number*
-*number* (Obsolescent)
Specifies the minimum string length. *number* is a positive decimal integer. The default is 4.

-o
(Obsolescent) Causes each string to be preceded by its offset in the file (in octal). This is equivalent to -t o.

-t *format* Prints the offset in the *format* specified. *format* can be one of the following:

- d Decimal
- o Octal
- x Hexadecimal

files The path name of a regular file to be used as input. If no *file* operand is specified, input is read from the standard input.

The `strings` utility is useful for identifying random object files.

NOTES

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

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

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

EXIT STATUS

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

0 Successful completion.

>0 An error occurred.

BUGS

The algorithm for identifying strings is extremely primitive. In particular, machine code instructions on certain architectures can resemble sequences of ASCII bytes, which will fool the algorithm.

SEE ALSO

`od`(1)

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

NAME

`strip` – Removes symbol table from an executable file

SYNOPSIS

`strip` [`-s`] [`-V`] *files*

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4
CRI extensions (`-s` and `-V` options)

DESCRIPTION

The `strip` utility removes the symbol table information from executable files. After this has been done, no symbolic debugging access is available from files.

This is useful for saving disk space after a program has been debugged (there is no effect on memory used at run time by the program). The effect of `strip` is the same as using the `-s` option on `segldr`(1) or `ld`(1).

The `strip` utility accepts the following options:

`-s` Prints to standard output a summary of the files that have been successfully stripped. The summary includes the name of the file, the original size of the file in bytes, the size of the file in bytes with the symbol table removed, the size in bytes of the symbol table removed, and the percentage of the original file that consisted of symbol tables.

`-V` Outputs the `strip` version number to standard error.

files Specifies the files to be stripped.

NOTES

A stripped executable file should not be used for profiling with `prof`(1) and cannot be used with debuggers.

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

Active Category	Action
`system`, `secadm`	Allowed to remove the symbol table from any executable file. In a privileged administrator shell environment, shell-redirected I/O is not subject to file protections.
`sysadm`	Allowed to remove the symbol table from any executable file subject to security label restrictions. Shell-redirected I/O is subject to security label restrictions.

If the PRIV_SU configuration option is enabled, the super user is allowed to remove the symbol table from any executable file. Shell-redirected I/O on behalf of the super user is not subject to file protections.

EXIT STATUS

The exit status matches the number of files that failed to have their symbol tables removed.

FILES

a.out

SEE ALSO

ar(1)
ld(1) to invoke the link editor
prof(1) to show where execution time is spent
segldr(1) to invoke the CRI segment loader (SEGLDR)

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

NAME

stty – Sets the options for a terminal

SYNOPSIS

```
stty -a
stty -g
stty [options]
```

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The stty utility sets certain terminal I/O options for the device that is the current standard input; without arguments, it reports the settings of certain options.

In this report, if a character is preceded by a caret (^), the value of that option is the corresponding CONTROL character (for example, ^h is <CONTROL-h>; in this case, recall that <CONTROL-h> is the same as the backspace key.) The sequence ^´ or ^@ means that an option has a null value.

The stty utility accepts the following options:

-a Writes to standard output all of the current settings for the terminal.

-g Writes to standard output all of the current settings in a form that can be used as an argument to another stty utility. (See Examples 4 and 5 in the EXAMPLES section.)

For detailed information about the modes listed the Control Modes through Local Modes subsections, see termio(4). Options in the last group are implemented with options in the previous groups. Many combinations of options make no sense, but no sanity checking is performed. The options are selected from the following.

Control Modes

parenb (-parenb)	Enables (disables) parity generation and detection.
parext (-parext)	Enables (disables) extended parity generation and detection for mark and space parity.
parodd (-parodd)	Selects odd (even) parity.
cs5 cs6 cs7 cs8	Selects character size (see termio(4)).
0	Hangs up phone line immediately.

`110 300 600 1200 1800 2400 4800 9600 19200 38400`
Sets terminal baud rate to the number given if possible. All hardware interfaces do not support all speeds.

`ispeed 0 110 300 600 1200 1800 2400 4800 9600 19200 38400`
Sets terminal input baud rate to the number given. (Not all hardware supports split baud rates.) If the input baud rate is set to zero, the input baud rate is specified by the value of the output baud rate.

`ospeed 0 110 300 600 1200 1800 2400 4800 9600 19200 38400`
Sets terminal output baud rate to the number given, if possible. (Not all hardware supports split baud rates.) If the output baud rate is set to 0, the line will be hung up immediately.

`hupcl` (`-hupcl`) Hangs up (does not hang up) dataphone dataset connection on last close.

`hup` (`-hup`) Same as `hupcl` (`-hupcl`).

`cstopb` (`-cstopb`) Uses two (one) stop bits per character.

`cread` (`-cread`) Enables (disables) the receiver.

`clocal` (`-clocal`) Assumes a line without (with) modem control.

`loblk` (`-loblk`) Blocks (does not block) output from a noncurrent layer.

Input Modes

`ignbrk` (`-ignbrk`) Ignores (does not ignore) break on input.

`brkint` (`-brkint`) Signals (does not signal) interrupt on break.

`ignpar` (`-ignpar`) Ignores (does not ignore) parity errors.

`parmrk` (`-parmrk`) Marks (does not mark) parity errors. See `termio`(4).

`inpck` (`-inpck`) Enables (disables) input parity checking.

`istrip` (`-istrip`) Strips (does not strip) input characters to seven bits.

`inlcr` (`-inlcr`) Maps (does not map) `<newline>` to `<return>` on input.

`igncr` (`-igncr`) Ignores (does not ignore) CR on input.

`icrnl` (`-icrnl`) Maps (does not map) CR to NL on input.

`iuclc` (`-iuclc`) Maps (does not map) uppercase alphabetic characters to lowercase on input.

`ixon` (`-ixon`) Enables (disables) START/STOP output control. Output is stopped by sending an ASCII DC3 and started by sending an ASCII DC1.

`ixany` (`-ixany`) Allows any character (only DC1) to restart output.

`ixoff` (`-ixoff`) Requests that the system send (not send) START/STOP characters when the input queue is nearly empty or full.

`imaxbel` (`-imaxbel`)
: Echoes (does not echo) BEL when the input line is too long.

Output Modes

`opost` (`-opost`)
: Post-processes output (does not post-process output; ignores all other output modes).

`olcuc` (`-olcuc`)
: Maps (does not map) lowercase alphabetic characters to uppercase on output.

`onlcr` (`-onlcr`)
: Maps (does not map) NL to CR-NL on output.

`ocrnl` (`-ocrnl`)
: Maps (does not map) CR to NL on output.

`onocr` (`-onocr`)
: Does not (does) output CRs at column 0.

`onlret` (`-onlret`)
: On the terminal, NL performs (does not perform) the CR function.

`ofill` (`-ofill`)
: Uses fill characters (use timing) for delays.

`ofdel` (`-ofdel`)
: Uses delete characters (NULs) as fill characters.

`cr0 cr1 cr2 cr3`
: Selects style of delay for carriage returns. See `termio`(4).

`nl0 nl1`
: Selects style of delay for line-feeds. See `termio`(4).

`tab0 tab1 tab2 tab3`
: Selects style of delay for horizontal tabs. See `termio`(4).

`bs0 bs1`
: Selects style of delay for backspaces. See `termio`(4).

`ff0 ff1`
: Selects style of delay for form-feeds. See `termio`(4).

`vt0 vt1`
: Selects style of delay for vertical tabs. See `termio`(4).

Local Modes

`extproc` (`-extproc`)
: Enables (disables) external processing mode. When in external processing mode, much of the functionality of the tty driver is omitted (for example, line editing and echoing of typed data), because it is assumed that this functionality is being done externally. This is the mode that `telnetd`(8) uses when it is running with the Telnet Linemode option enabled. Disabling external processing notifies the `telnetd`(8) process, which then disables the line-mode option.

`isig` (`-isig`)
: Enables (disables) the checking of characters against the special control characters INTR, QUIT, and SWTCH.

`icanon` (`-icanon`)
: Enables (disables) canonical input (ERASE and KILL processing).

`xcase` (`-xcase`)
: Canonical (unprocessed) uppercase and lowercase presentation.

`echo` (`-echo`)
: Echoes back (does not echo back) every character typed.

`echoe` (`-echoe`)	Echoes (does not echo) ERASE character as a backspace-space-backspace string. Note: This mode erases the ERASEed character on many CRT terminals; however, it does not keep track of column position and, therefore, may be confusing on escaped characters, tabs, and backspaces.
`echok` (`-echok`)	Echoes (does not echo) NL after KILL character.
`lfkc` (`-lfkc`)	The same as `echok` (`-echok`); obsolete.
`echonl` (`-echonl`)	Echoes (does not echo) NL.
`noflsh` (`-noflsh`)	Disables (enables) flush after INTR, QUIT, or SWTCH.
`stwrap` (`-stwrap`)	Disables (enables) truncation of lines longer than 79 characters on a synchronous line.
`tostop` (`-tostop`)	Sends (does not send) SIGTTOU when background processes write to the terminal.
`echoctl` (`-echoctl`)	Echoes (does not echo) control characters as *char*, delete as ^?
`echoprt` (`-echoprt`)	Echoes (does not echo) erase character as character is *erased*.
`echoke` (`-echoke`)	BS-SP-BS erase (does not BS-SP-BS erase) entire line on line kill. Note: The `echoke` mode is not available in UNICOS.
`flusho` (`-flusho`)	Output is (is not) being flushed.
`pendin` (`-pendin`)	Retypes (does not retype) pending input at next read or input character.
`iexten` (`-iexten`)	Enables (disables) extended (implementation-defined) functions for input data.
`stflush` (`-stflush`)	Enables (disables) flush on a synchronous line after every `write`(2) system call.
`stappl` (`-stappl`)	Uses application mode (uses line mode) on a synchronous line.

Control Assignments

control-character c	Sets *control-character* to *c*; *control-character* is `ctab`, `discard`, `dsusp`, `eof`, `eol`, `eol2`, `erase`, `intr`, `kill`, `lnext`, `quit`, `reprint`, `start`, `stop`, `susp`, `swtch`, or `werase`. `ctab` is used with `-stappl`; see `termio`(4). If *c* is preceded by an (escaped from the shell) caret (^), the value used is the corresponding `CONTROL` character (for example, `^d` is `<CONTROL-d>`); `^?` is interpreted as `<DEL>`, and `^-` and `undef` are interpreted as {_POSIX_VDISABLE} if {_POSIX_VDISABLE} is in effect for the terminal.
`min`, `time` *number*	Sets the value of `min` or `time` to *number*. `min` and `time` are used in noncanonical mode input processing (`-icanon`).
`line` *i*	Sets line discipline to *i* ($0 < i < 127$).

Combination Modes

`evenp` or `parity`	Enables `parenb` and `cs7`.
`oddp`	Enables `parenb`, `cs7`, and `parodd`.
`-parity` or `-evenp`	Disables `parenb`, and set `cs8`.
`-oddp`	Disables `parenb` and `parodd`, and set `cs8`.
`-spacep`	Disables `parenb` and `parext`, and set `cs8`.
`-markp`	Disables `parenb`, `parodd`, and `parext`, and set `cs8`.
`raw` (`-raw` or `cooked`)	Enables (disables) raw input and output (no ERASE, KILL, INTR, QUIT, SWTCH, EOT, or output postprocessing).
`nl` (`-nl`)	Sets (unsets) `icrnl`. In addition, `-nl` unsets `inlcr` and `igncr`.
`lcase` (`-lcase`)	Sets (unsets) `xcase`, `iuclc`, and `olcuc`.
`LCASE` (`-LCASE`)	Same as `lcase` (`-lcase`).
`tabs` (`-tabs` or `tab3`)	Preserves (expands to spaces) tabs when printing.
`ek`	Resets ERASE and KILL characters back to `<CONTROL-u>` and `<DEL>`.
`sane`	Resets all modes to some reasonable values.

Window Size

`rows` *n*	Sets window size to *n* rows.
`columns` *n*	Sets window size to *n* columns.
`ypixels` *n*	Sets vertical window size to *n* pixels.
`xpixels` *n*	Sets horizontal window size to *n* pixels.

NOTES

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

Active Category	Action
`system`, `secadm`	In a privileged administrator shell environment, allowed to write shell-redirected output to any file.
`sysadm`	Shell-redirected output is subject to security label restrictions.

If the `PRIV_SU` configuration option is enabled, the super user can write shell-redirected output to any file.

EXIT STATUS

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

0 The terminal options were read or set successfully.

>0 An error occurred.

EXAMPLES

Example 1: The following example reports certain terminal settings for the standard input device:

```
$ stty
speed 9600 baud; -parity hupcl -cread
rows = 40; columns = 80; ypixels = 0; xpixels = 0;
-inpck -istrip icrnl -ixany onlcr tab3
extproc echo echoe echok
```

Example 2: The following example reports all terminal settings for `/dev/tty`:

```
$ stty -a </dev/tty
speed 9600 baud; line = 0;
rows = 40; columns = 80; ypixels = 0; xpixels = 0;
intr = ^c; quit = ^; erase = ^?; kill = ^u;
eof = ^d; eol = ^@; eol2 = ^@; swtch = ^z;
start = ^q; stop = ^s; susp = ^z; dsusp = ^y;
rprnt = ^r; flush = ^o; werase = ^w; lnext = ^v;
-parenb -parodd cs8 -cstopb hupcl -cread -clocal -loblk
-ignbrk -brkint ignpar -parmrk -inpck -istrip -inlcr -igncr icrnl -iuclc
ixon -ixany -ixoff
extproc isig icanon -xcase echo echoe echok -echonl -noflsh
-tostop -iexten
opost -olcuc onlcr -ocrnl -onocr -onlret -ofill -ofdel tab3
```

Example 3: The following example turns off character echo on the current terminal:

```
$ stty -echo
$
```

Example 4: The following example reports the current settings for use on a later `stty` command line:

```
$ stty -g
d20:1805:4bd:13b:3:1c:8:15:4:0:0:1a
$
```

Example 5: The following example shows how to use the output of a previous `stty -g` command to set the current terminal:

```
$ stty d20:1805:4bd:13b:3:1c:8:15:4:0:0:1a
$
```

Example 6: The following example sets the current terminal characteristics to those of `/dev/ttyp003`:

```
$ stty ‘stty -g </dev/ttyp003‘
```

SEE ALSO

`ioctl`(2) to perform functions on character special files in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

`termio`(4) for information on general terminal interface in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

`telnetd`(8) to invoke the DARPA TELNET protocol server in the *UNICOS Administrator Commands Reference Manual*, Cray Research publication SR–2022

NAME

su – Lets you become another user or the super user

SYNOPSIS

su [-] [*name* [*args*]]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The su utility allows you to become another user without logging off. The default user *name* is root.

To use the su utility, the appropriate password must be supplied (unless you are an appropriately authorized user). If the password is correct, su will execute a new shell with the real and effective user ID set to that of the specified user. The new shell will be the optional program specified in the shell field of the specified user's password file entry (see udb(5)), or /bin/sh if none is specified (see sh(1)). To restore your original user identity, exit the new shell.

Any additional arguments specified on the command line are passed to the program invoked as the shell. For example, when sh(1) is used, an argument of the form -c *string* executes *string* via the shell and an option of -r will give the user a restricted shell.

The following statements are true only if the optional program specified in the shell field of the specified user's password file entry is like sh(1). If the first argument to su is -, the environment will be changed to what would be expected if the user actually logged in as the specified user. This is done by invoking the program used as the shell with an *arg0* value whose first character is -, causing the system's profile (/etc/profile) and then the specified user's profile (.profile in the new home directory) to be executed. Otherwise, the environment is passed along, with the possible exception of $PATH, which is set to /bin:/etc:/usr/bin for root. If the optional program used as the shell is /bin/sh, the user's .profile can check *arg0* for -sh or -su to determine whether it was invoked by login(1) or su, respectively. If the user's program is not /bin/sh, the program is invoked with an *arg0* of -*program* by both login(1) and su.

All attempts to become another user using su are logged in the log file /usr/adm/sulog. In addition, you are limited to two failed su attempts per minute. You are cautioned after the second failure, and logged off and disabled from relogging on after the third failure in any minute, and the setuid(2) system calls are logged in the security log.

The su utility invokes the centralized identification and authorization library routines to validate the user ID and password.

The su utility accepts the following options:

- Changes environment to that of specified user name.

name Indicates user name to which to log on (default is root).

args Specifies shell arguments for new login.

NOTES

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

Privilege Text	Action
`all`	Allowed to `su` to any user without supplying a password.

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

Active Category	Action
`system`, `secadm`	Allowed to `su` to any user without supplying a password.

If the `PRIV_SU` configuration option is enabled, the super user is allowed to `su` to any user without password.

CAUTIONS

If the `-` argument is used, the `TMPDIR` environment variable for the specified user is set to `JTMPDIR`, which is defined in `/usr/include/tmpdir.h`.

New limits are not set for the new user; the current limits are inherited.

EXAMPLES

Example 1: To become user `bin` while retaining your previously exported environment, enter the following:

```
su bin
```

Example 2: To become user `bin` but change the environment to what would be expected if `bin` had originally logged in, enter the following:

```
su - bin
```

Example 3: To execute *command* with the temporary environment and permissions of user `bin`, enter the following:

```
su - bin -c "command  arguments"
```

FILES

`/etc/udb`	User validation file containing user control limits
`/etc/profile`	System's start-up file for standard shell
`$HOME/.profile`	User's start-up file for standard shell
`/usr/adm/sulog`	Log file

SEE ALSO

`env`(1), `login`(1), `privtext`(1), `sh`(1), `tmpdir`(1)

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

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

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

General UNICOS System Administration, Cray Research publication SG–2301

NAME

sum – Prints checksum and block count of a file

SYNOPSIS

sum [-r] [*files*]

IMPLEMENTATION

All Cray Research systems

STANDARDS

XPG4

DESCRIPTION

The sum utility calculates and writes a 16-bit checksum for *files* and writes the number of 512-byte blocks in the files to the standard output. If no files are specified, data is read from standard input.

The sum utility accepts the following option and operand:

-r Causes an alternative algorithm to be used in computing the checksum.

files Specifies the files to be checked.

NOTES

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

Active Category	Action
system, secadm	Allowed to print checksum and size information for any file. In a privileged administrator shell environment, shell-redirected I/O is not subject to file protections.
sysadm	Allowed to print checksum and size information for any file subject to security label restrictions. Shell-redirected I/O is subject to security label restrictions.

If the PRIV_SU configuration option is enabled, the super user is allowed to print checksum and size information for any file. Shell-redirected I/O on behalf of the super user is not subject to file protections.

EXIT STATUS

The sum utility exits with one of the following values:

0 Successful completion.

>0 An error occurred.

MESSAGES

`Read error` Indistinguishable from end-of-file on most devices; check the block count.

SEE ALSO

`cksum`(1), `wc`(1)

NAME

sync – Flushes file system cache

SYNOPSIS

```
sync
```

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The sync utility executes the sync system primitive. It flushes all previously unwritten system buffers out of main memory. The buffers are written to disk unless a logical device cache is being used. In this instance, buffers are written to a solid-state storage device (SSD) and an ldsync(8) utility is required to flush the buffers to disk. If the system is to be stopped, sync(2) and ldsync(8) must be called to ensure file system integrity. See sync(2) for details.

SEE ALSO

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

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

NAME

sysconf – Displays system configuration data

SYNOPSIS

```
sysconf
```

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The sysconf utility displays hardware and software configuration data available from the sysconf(2) system call. The display uses the last part of the system call parameter as a keyword description, so that information as to the meaning of a field can be easily looked up in the sysconf(2) description. For example, the parameter _SC_CRAY_SYSTEM is displayed as SYSTEM= ???.

NOTES

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

Active Category	Action
system, secadm	In a privileged administrator shell environment, allowed to write shell-redirected output to any file.
sysadm	Shell-redirected output is subject to security label restrictions.

If the PRIV_SU configuration option is enabled, the super user can write shell-redirected output to any file.

SEE ALSO

getconf(1)

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

NAME

tabs – Sets tabs on a terminal

SYNOPSIS

tabs [-T *term*] [-*n* | -a | -a2 | -c | -c2 | -c3 | -f | -p | -s | -u]

tabs [-T *term*] *n1*[,*n2*,...]

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The tabs utility sets the tab stops on the user's terminal according to the tab specification *tabspec*, after clearing any previous settings. The user's terminal must have hardware tabs that can be set remotely. The tabs utility accepts the following options:

Two types of tab specification are accepted for *tabspec*, either repetitive (-*n*) or arbitrary (*n1*,*n2*,...). If you omit *tabspec*, the default value is -8, (for example, UNIX system ''standard'' tabs). The lowest column number is 1. For tabs, column 1 always refers to the leftmost column on a terminal, even if column markers begin at 0.

-*n* A *repetitive* specification requests tabs at columns 1+ *n*, 1+2* *n*, and so on. The value 8 represents the UNIX system ''standard'' tab setting, and it is the most likely tab setting to be found at a terminal. The value 0 implies no tabs.

n1[, *n2*,...]
The *arbitrary* format permits the user to type any chosen set of numbers, separated by commas, in ascending order. Up to 40 numbers are allowed. If any number (except the first one) is preceded by a plus sign, it is considered an increment to be added to the previous value. Thus, the formats 1,10,20,30, and 1,10,+10,+10 are considered identical.

-T *term* The tabs utility usually must know the type of terminal to set tabs. *term* is a name listed in term(5). If you omit the -T option, tabs uses the value of the TERM environment variable. If TERM is not defined in the environment (see environ(7)), tabs tries a sequence that will work for GE Terminet 300 terminals.

The following options set tabs to some commonly used values:

-a 1,10,16,36,72
Assembler, format one.

-a2 1,10,16,40,72
Assembler, format two.

`-c` 1,8,12,16,20,55
COBOL, normal format.

`-c2` 1,6,10,14,49
COBOL, compact format one.

`-c3` 1,6,10,14,18,22,26,30,34,38,42,46,50,54,58,62,67
COBOL, compact format two.

`-f` 1,7,11,15,19,23
FORTRAN.

`-p` 1,5,9,13,17,21,25,29,33,37,41,45,49,53,57,61
PL/1.

`-s` 1,10,55
SNOBOL.

`-u` 1,12,20,44
Alternative assembler format.

To set tabs, use standard output.

NOTES

No consistency exists among different terminals to clear tabs.

This implementation relies entirely on information in the `terminfo` database for its operation.

EXIT STATUS

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

0 Successful completion.

>0 An error occurred.

SEE ALSO

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

`environ`(7) (available only online)

NAME

`tail` – Copies the last part of a file

SYNOPSIS

`tail` [`-f`] [`-c` *number*] [*file*]
`tail` [`-f`] [`-b` *number*] [*file*]
`tail` [`-f`] [`-n` *number*] [`-r`] [*file*]

Obsolescent version; may not be supported in future releases:

`tail` [±[*number*][*unit*][`r`][`f`]] [*file*]

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4
AT&T extensions (`-b` and `-r` options)

DESCRIPTION

The `tail` utility copies *file* to standard output, beginning at a designated place. If you omit *file*, standard input is used.

The `tail` utility accepts the following options:

`-f` With this option, if the input file is not a pipe, the program will not terminate after the line of the input file has been copied, but will enter an endless loop. It sleeps for a second and then attempts to read and copy more records from the input file. Thus, you can use it to monitor the growth of a file that is being written by some other process. If you omit *file*, and standard input is a pipe, this option is ignored.

`-c` *number* The *number* argument is a decimal integer whose sign affects the location in the file, measured in bytes, to begin the copying. If the sign is `+`, copying starts relative to the beginning of the file. If the sign is `-` or if you omit the sign, copying starts relative to the end of the file. The origin for counting is 1 (for example, `-c +1` represents the first byte of the file, `-c -1` the last byte of the file).

`-b` *number* This option is equivalent to `-c` *number*, except the starting location in the file is measured in 512-byte blocks rather than bytes.

`-n` *number* This option is equivalent to `-c` *number*, except the starting location in the file is measured in lines rather than bytes.

`-r` This option copies lines from the end of the file in reverse order. The default is to print the entire file in reverse order.

file File to be copied to standard output.

In the obsolescent version, an argument that begins with a `-` or `+` can be used as a single option. *unit* can be one of `b`, `c`, or `l`. The ±*number* argument with the letter `c` specified as a suffix is equivalent to `-c` ±*number*; ±*number* with the letter `l` specified as a suffix is equivalent to `-n` ±*number*; ±*number* with the letter `b` specified as a suffix is equivalent to `-b` ±*number*. If you omit *unit*, `n` is assumed. If you omit *number*, 10 is used. The letter `f` specified as a suffix is equivalent to specifying the `-f` option. Specifying the letter `r` alone or with the `l` suffix is equivalent to `-r`.

In the nonobsolescent form, if you omit `-b`, `-c`, or `-n`, `-n 10` is assumed.

NOTES

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

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

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

CAUTIONS

The `tail` utility copies only the last 32,768 bytes of a file, regardless of its line count.

EXIT STATUS

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

0 Successful completion.

>0 An error occurred.

BUGS

Because `tail` commands that are relative to the end of the file are stored in a buffer, they are limited in length.

Various kinds of anomalous behavior may happen with character special files.

EXAMPLES

Example 1: The following command prints the last 10 lines of the file `fred`, followed by any lines that are appended to `fred` between the time `tail` is initiated and killed:

```
tail -f fred
```

Example 2: The following command prints the last 15 characters of the file `fred`, followed by any lines that are appended to `fred` between the time `tail` is initiated and killed:

```
tail -c 15 -f fred
```

SEE ALSO

`cat`(1), `head`(1), `more`(1), `pg`(1)

NAME

`talk` – Enables one user to communicate with another user

SYNOPSIS

`/usr/ucb/talk` *address* [*terminal*]

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The `talk` utility is a visual communication program that copies lines from your terminal to that of another user. This utility requires the TCP/IP networking software running under UNICOS.

The `talk` utility accepts the following operands:

address Specifies the login name of the person to whom you want to talk. If you want to talk to someone on your own system, use the person's login name for the *address* argument. If you want to talk to someone on a different host system, use one of the following formats for *address*:

user@host (preferred usage)
host!user
host:user

terminal Indicates the terminal name. If you want to talk to a user who is logged in more than once, use the *terminal* argument to indicate the appropriate terminal name.

When first called, the `talk` utility sends the following message to the person to whom you want to talk:

```
Message from TalkDaemon@his_machine...
talk: connection requested by your_name@your_machine.
talk: respond with: talk your_name@your_machine
```

At this point, the message recipient should reply by entering the following:

```
talk your_name@your_machine
```

It does not matter from which machine the recipient replies, as long as the login name is the same. When communication is established, the two parties may type simultaneously, with their output appearing in separate windows. If you press <CONTROL-l>, the screen is reprinted; the erase, kill, and word-kill characters work normally. If you press <CONTROL-g>, an <alert> character is sent to both terminals. To exit the talk utility, type an interrupt character; the cursor moves to the bottom of the screen and the command restores the terminal.

Users may deny or grant other users the permission to talk to them by using the mesg(1) command. At the outset, the use of talk is allowed. Certain utilities, in particular pr(1), disallow messages to prevent messy output.

NOTES

The talk utility requires the TCP/IP networking software to run under UNICOS.

EXIT STATUS

The talk utility exits with one of the following values:

0 Successful completion.

>0 An error occurred, or talk was invoked on a terminal incapable of supporting it.

FILES

/etc/hosts	Finds the recipient's machine
/etc/utmp	Finds the recipient's tty

SEE ALSO

mail(1), mesg(1), pr(1), who(1), write(1)

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

NAME

`tar` – Archives tape files

SYNOPSIS

`tar -c`[*modifiers*] [*files*]
`tar -r`[*modifiers*] [*files*]
`tar -t`[*modifiers*] [*files*]
`tar -u`[*modifiers*] [*files*]
`tar -x`[*modifiers*] [*files*]

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The `tar` utility saves and restores files on magnetic tape and disk files.

Note: The `tar` options can be followed by zero or more *modifiers*. For descriptions of the *modifiers*, see the subsection following the option descriptions.

The `tar` utility supports the following options:

`-c` (Create) Creates a new archive; writing begins at the beginning of the archive, instead of after the last file.

`-r` (Replace) Writes the specified *files* on the end of the archive. This option does not work with tape devices.

`-t` (Table) Lists the names and other information for the specified files each time that they occur on the archive. The listing is similar to the format produced by the `ls -l` command. If you omit *files*, all the names on the archive are listed.

`-u` (Update) Adds the specified *files* to the archive if they are not already there or have been modified since last written on that archive. This option implies the `-r` option.

`-x` (Extract) Extracts the specified *files* from the archive. If a specified file matches a directory whose contents had been written onto the archive, this directory is (recursively) extracted. You must use the file or directory's relative path when appropriate; otherwise `tar` will not find a match. The owner, modification time, and mode are restored (if possible). If you omit *files*, the entire content of the archive is extracted. If several files with the same name are on the archive, the last one overwrites all earlier ones.

The tar options can be followed immediately by zero or more of the following *modifiers*:

a Excludes copy or preservation of access control lists (ACLs). The a modifier is useful only with the s modifier.

b (Blocking factor) Causes tar to use the *block* argument as the blocking factor for (tape) records in the archive. The default is 20 for tape archives and 128 for disk archives. The maximum is 128. The size of a block is 512 bytes. The block size is determined automatically when reading tapes created on block special devices (-x and -t options). The blocking factor should match the argument given to the -b option of the tpmnt(1) utility for archives written to tape.

f (File) Causes tar to use the next argument as the name of the archive. If you omit the f modifier, tar uses the default, which is /dev/extape. If the name of the file is -, tar writes to the standard output or reads from the standard input, whichever is appropriate. Thus, you can use tar as the head or tail of a pipeline. You can also use tar to move hierarchies with the utility, as follows:

```
cd fromdir; tar -cf - . | (cd todir; tar -xf -)
```

h Follows symbolic links as if they were normal files or directories. Usually, tar does not follow symbolic links.

l (Link) tar sends an error message if it cannot resolve all the links to the files being dumped. If you omit the l modifier, no error messages are printed.

m (Modify) tar does not restore the modification times. If you use the m modifier, the modification time of the file will be the time of extraction. The m modifier is valid only with the -x option.

o (Ownership) Causes extracted files to take on the user and group identifier of the user running the program, rather than those on tape. This option is always on, unless the O modifier is used, and is valid only with the -x option.

O (Ownership) Turns off the o modifier. This causes extracted files to take on the user and group identifier from the tape, rather than those of the user running the program. Users must have chown(2) permission in their UDB record to use this option.

p (Permissions) Preserves the original file permissions on extracted files. Clears the umask of the process that extracts the files. The p modifier applies only to the -x option.

s (Secure) Performs a secure copy (security information and ACLs).

v (Verbose) Usually, tar does its work silently. The v modifier causes tar to display the name of each file it treats, preceded by the option. With the -t option, the v modifier gives more information about the tape entries than the ls(1) command.

w (What) Causes tar to display the action to be taken, followed by the name of the file, and then to wait for your confirmation. If you begin a word with y, the action is performed. Any other input means ‘‘no.’’ The w modifier is not valid with the -t option.

The `tar` utility supports the following operand:

files Specifies which files (or directories) are to be dumped or restored. In all cases, appearance of a directory name refers to the files and (recursively) subdirectories of that directory.

A secure save (`s` modifier) outputs security information and access control lists (ACLs) corresponding to regular files. A secure restore (`s` modifier) installs security information and ACLs of regular files.

NOTES

By default, when extracting files, the modes of the files will be set using the present `umask`. To preserve the original modes, you can invoke `tar` using the `p` modifier.

You may copy only files that have the same security label as the user.

A user's active categories must dominate the categories of the file being copied.

The `tar` utility saves information about regular files. Thus, security information and ACLs for directories are not preserved.

ACLs are preserved only if the user is the owner of the file, or has an active `secadm` category, or has both read and write access to the file being copied.

CAUTIONS

It is recommended that you use a blocking factor of 8 on all tapes. For example, the following creates a tape with a blocking factor of 8:

```
tar -cb 8
```

MESSAGES

Reports non-valid options and tape read and write errors.
Reports not enough memory available to hold the link tables.

BUGS

You cannot request the *n*th occurrence of a file.

The `-u` option can be slow.

You must not use the `b` modifier with archives that will be updated. The current magnetic tape driver cannot backspace raw magnetic tape. If the archive is on a disk file, you must not use the `b` modifier, because updating an archive stored on disk can destroy it.

The length of a file name is currently limited to 100 characters.

The `tar` utility does not copy empty directories or special files.

EXAMPLES

Example 1: The following command backs up a user's entire directory to online magnetic tape:

```
cd
rsv CART 1
tpmnt -l nl -v vsn -P tapefile -b 4096 -g CART -n
tar -cvfb tapefile 8 .
rls -a
```

The *vsn* variable is the volume serial number of the tape. *tapefile* is the path name for the tape being used. A blocking factor of 8 (4096 bytes) was chosen so that the tape can be read back with online tape.

Example 2: The following example demonstrates how you can use `tar` to write and then read an online tape file of blocking factors other than 8:

```
rsv
tpmnt -l nl -p tapefile -b 512 -v vsn
tar -cvfb tapefile 1 .
cd newdir
tar -xvf tapefile
rls -a
```

Example 3: This example shows how to read a `tar` tape from another UNIX system that was written with a blocking factor of 20:

```
rs
tpmnt -l nl -p tapefile -b 10240 -v vsn
tar -xvf tapefile
rls -a
```

FILES

`/dev/extape`	External tape file
`/tmp/tar*`	Work files

SEE ALSO

`ar`(1), `cpio`(1), `ls`(1), `rls`(1), `rsv`(1), `tpmnt`(1)

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

General UNICOS System Administration, Cray Research publication SG–2301

NAME

`target` – Verifies target CPU characteristics

SYNOPSIS

`target` [`-s`] [*cpuname*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `target` utility determines and prints the CPU characteristic for the machine type specified by *cpuname*.

The `target` utility accepts the following options:

`-s` Prints only the machine subtype field. The machine subtype field may have one of the following values:

Machine Subtype	Description
`CRAY-C90`	CRAY C90 series
`CRAY-EL`	CRAY EL series
`CRAY-J90`	CRAY J90 series
`CRAY-JSE`	CRAY J90se series
`CRAY-TS`	CRAY T90 series
`CRAY-TS-IEEE`	CRAY T90 series with IEEE floating point
`CRAY-T3D`	CRAY T3D systems
`CRAY-T3E`	CRAY T3E systems
`CRAY-YMP`	CRAY Y-MP model 832

If the `-s` option is not specified, the `target` utility prints the subtype field without the `CRAY-` prefix. This is printed when the target is a Cray PVP system.

cpuname Specifies a Cray Research machine type. The *cpuname* argument can be any one of the following (uppercase or lowercase):

Machine Type	Description
`cray-c90`	CRAY C90 series
`cray-el`	CRAY EL series
`cray-j90`	CRAY J90 series
`cray-jse`	CRAY J90se series
`cray-ts`	CRAY T90 series
`cray-t3d`	CRAY T3D systems
`cray-t3e`	CRAY T3E systems
`cray-ymp`	CRAY Y-MP model 832
`host`	
`\*host`	

```
iop
target          Default
\*target
```

If *cpuname* is `target`, the current environment is checked for the `TARGET` environment variable. If it is found, `target` parses the environment variable and displays the machine characteristics for the target machine. If *cpuname* is `host`, the host machine characteristics are displayed. For any other specified machine type, the machine type's default characteristics are given.

The `target` utility does not make changes to the current `TARGET` environment variable. The user must initialize and/or make changes to this environment variable at the shell level when necessary. The `target` utility verifies that the environment variable is syntactically correct.

To initialize the `TARGET` environment variable, enter the following in the standard shell:

```
export TARGET
```

The format to set up or change the `TARGET` environment variable in the standard shell is as follows:

```
TARGET=[cpuname]{,[charac]}
```

The target machine represented by `TARGET` takes on the default machine characteristics specified by *cpuname*, modified accordingly by any specified *charac* arguments. If you do not specify *cpuname*, the machine characteristics of the host machine are used and modified. On CRAY T3E systems, the characteristics of the PE on which the `target` utility is executing are returned.

The *cpuname* and *charac* arguments on the `TARGET` variable can be the following:

cpuname Same options as listed previously.

charac These are possible features that may be specified for the given *cpuname* computer. In some cases, it is possible to choose characteristics that may not make sense for a given *cpuname* computer. You cannot specify characteristics for `iop`.

All Cray Research systems let you specify the following numerical trait:

Numeric Trait	**Description**
`memsize=`*n*[*y*]	Memory size in words. Using `k` for *y* defines (*n* * 1024) words, using `m` for *y* defines (*n* * 1,048,576) words.

Cray PVP systems also let you specify the following numerical traits:

Numeric Trait	**Description**
`banks=`*n*	Number of memory banks.
`numcpus=`*n*	Number of CPUs.
`ibufsize=`*n*	Instruction buffer size.
`memspeed=`*n*	Memory speed in clock periods.
`clocktim=`*n*	Clock period in picoseconds.

`numclstr=`*n* Number of clusters.

`bankbusy=`*n* Number of clock periods that the memory bank reserved.

Cray PVP systems (except CRAY T90 series where noted) let you specify the following logical traits:

Logical Trait	Description
`bmm`	Bit matrix multiply unit.
`nobmm`	No bit matrix multiply unit (not valid for `cray-ts`).
`ema`	Extended memory addressing for 24-bit mode (not valid for `cray-ts`).
`noema`	No extended memory addressing for 24-bit mode.
`cigs`	Compressed index and gather/scatter.
`nocigs`	No compressed index or gather/scatter.
`vpop`	Vector pop count.
`novpop`	No vector pop count (not valid for `cray-ts`).
`pc`	Programmable clock.
`nopc`	No programmable clock (not valid for `cray-ts`).
`readvl`	Read vector length.
`noreadvl`	Do not read vector length (not valid for `cray-ts`).
`vrecur`	Vector recursion (not valid for `cray-ts`).
`novrecur`	No vector recursion.
`avl`	Additional vector logical.
`noavl`	No additional vector logical (not valid for `cray-ts`).
`hpm`	Hardware performance monitor.
`nohpm`	No hardware performance monitor (not valid for `cray-ts`).
`statrg`	Status register.
`nostatrg`	No status register (not valid for `cray-ts`).
`bdm`	Bidirectional memory.
`nobdm`	No bidirectional memory (not valid for `cray-ts`).
`cori`	Control operand range interrupts.
`nocori`	No control operand range interrupts (not valid for `cray-ts`).
`addr32`	32-bit mode addressing (not valid for `cray-ts`).
`noaddr32`	No 32-bit mode addressing.

`xea`	CRAY Y-MP instruction timings (not valid for `cray-ts`).
`noxea`	No CRAY Y-MP instruction timings.
`avpop`	Additional vector pop count.
`noavpop`	No additional vector pop count.
`ieee`	IEEE floating-point arithmetic (not valid for `cray-ymp` or `cray-c90`).
`noieee`	No IEEE floating-point arithmetic.

NOTES

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

Active Category	Action
`system`, `secadm`	In a privileged administrator shell environment, allowed to write shell-redirected output to any file.
`sysadm`	Shell-redirected output is subject to security label restrictions.

If the `PRIV_SU` configuration option is enabled, the super user can write shell-redirected output to any file.

SEE ALSO

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

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

`GETPMC`(3F) in the *Application Programmer's Library Reference Manual*, Cray Research publication SR–2165

NAME

tbl – Formats tables for nroff(1) or troff(1)

SYNOPSIS

tbl [-TX] [*filename*] ...

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The tbl preprocessor prepares tables for nroff(1) or troff(1). tbl assumes that lines between the .TS and .TE command lines describe tables; thus they are reformatted. Lines outside these command lines are copied to the standard output. tbl does not alter the .TS and .TE command lines.

If no arguments are given, tbl reads the standard input, so tbl may be used as a filter. When tbl is used with eqn(1) or neqn(1), the tbl command should be first, to minimize the volume of data passed through pipes.

The tbl preprocessor accepts the following options:

-TX Forces tbl to use full vertical line motions. This option makes the output more suitable for devices that cannot generate partial vertical line motions (for example, line printers).

filename Specifies the files to be formatted.

The tbl preprocessor also accepts the following global options:

center Centers the table (default is left-adjust); expand makes the table as wide as the current line length.

box Encloses the table in a box.

doublebox Encloses the table in a double box.

allbox Encloses each item of the table in a box.

tab (*x*) Uses the character *x* instead of a tab to separate items in a line of input data.

linesize (*n*) Sets line or rules (for example, from box) in *n*-point type.

End the global options, if any, with a semicolon (;).

After global options come lines describing the format of each line of the table. Each such format line describes one line of the table itself, except that the last format line (which you must end with a period) describes *all* remaining lines of the table. A single key letter describes each column of each line of the table. You can follow this key letter with specifiers that determine the font and point size of the corresponding item, that indicate where vertical bars are to appear between columns, and that determine column width, intercolumn spacing, and so on. The available key letters are as follows:

c Centers item within the column.

r Right-justifies item within the column.

l Left-justifies an item within the column.

n Numerically adjusts item in the column: units positions of numbers are aligned vertically.

s Spans previous item on the left into this column.

a Centers longest line in this column and then left-justifies all other lines in this column with respect to that centered line.

^ Spans down previous entry in this column.

_ Replaces this entry with a horizontal line.

= Replaces this entry with a double horizontal line.

The characters B and I stand for the bold and italic fonts, respectively; the character | indicates a vertical line between columns.

The format lines are followed by lines containing the actual data for the table, followed finally by .TE. Within such data lines, data items are usually separated by tab characters.

If a data line consists of only _ or =, a single or double line, respectively, is drawn across the table at that point; if a *single item* in a data line consists of only _ or =, then that item is replaced by a single or double line. Some printers do not have the vertical resolution to produce double lines.

EXAMPLES

The following tbl example shows a simple three-column table. The characters \t represent a tab; when entering the text, type a genuine tab character:

```
.TS
c s s
c c s
c c c
l n n.
Household\tPopulation
Town\tHouseholds
\tNumber\tSize
Bedminster\t789\t3.26
Bernards Twp.\t3087\t3.74
Bernardsville\t2018\t3.30
Bound Brook\t3425\t3.04
Branchburg\t1644\t3.49
Bridgewater\t7897\t3.81
Far Hills\t240\t3.19
.TE
```

This input produces the following formatted table:

Town	Household Population	
	Households	
	Number	Size
Bedminster	789	3.26
Bernards Twp.	3087	3.74
Bernardsville	2018	3.30
Bound Brook	3425	3.04
Branchburg	1644	3.49
Bridgewater	7897	3.81
Far Hills	240	3.19

SEE ALSO

eqn(1), nroff(1), troff(1)