NAME

`ssd` – Display and change SSD configuration information

SYNOPSIS

`/etc/ssd` [`-a`] [`-c`] [`-e`] [`-p`] [`-q`] [`-R`] [`-u` *channel*] [`-d` *channel*] [`-t` *blocks*] [`-T` *blocks*]

IMPLEMENTATION

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

DESCRIPTION

The `ssd` command is used to display and modify SSD configuration information. The `ssd` command can be used to *up* or *down* SSD VHISP channels.

You must be an appropriately authorized user to execute the `ssd` command.

`-a`	Display all VHISP and SSD information. The output of this option is not meant to be very readable, it is used mainly for debugging.
`-c`	Display the VHISP and SSD configuration. See the EXAMPLES section.
`-e`	Display the VHISP and SSD error counter information. See the EXAMPLES section.
`-p`	Display the VHISP and SSD performance information. See the EXAMPLES section.
`-q`	Display the VHISP and SSD queuing information. See the EXAMPLES section.
`-R`	Reset all VHISP and SSD performance and queuing information to an initial state of all zeroes.
`-u` *channel*	*Up* the specified *channel*. This option makes the *channel* available for use by the system.
`-d` *channel*	*Down* the specified *channel*. This option makes the *channel* unavailable for use by the system.
`-t` *blocks*	Change the synchronous `ssd` threshold. The *blocks* argument is expressed in terms of 64-word blocks. SSD I/O transfers whose size is equal or less than this size will be done synchronously (*spin wait for channel interrupt*) within the low-level channel handler.
`-T` *blocks*	Change the synchronous `sds` threshold. The *blocks* argument is expressed in terms of 64-word blocks. SDS I/O transfers whose size is equal or less than this size will be done synchronously (*spin wait for channel interrupt*) within the low-level channel handler.

NOTES

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

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

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

EXAMPLES

Example 1: The following examples shows the `-c` option of the `ssd` command:

```
# /etc/ssd -c
Vhisps: 3 configured, 3 up, avail mask 07
  # indx chan(oct) avail
  0    0    1( 01)  up
  1    1    9(011)  up
  2    2   13(015)  up

SSDs: 2 configured
  # indx unit iopath hspmsk .size. ..bias.. ldthresh sdsthrsh maxtrans
  0    0    0     01     03    2Gw        0      200    10000   262136
  1    1    0     06     03    2Gw        0      200    10000   524272
```

Example 2: The following example shows the `-e` option of the `ssd` command:

```
# /etc/ssd -e
Vhisps: 3 configured, 3 up, avail mask 07
  # indx avail chan  errors recvrd unrcvd
  0    0     1    1       0      0      0
  1    1     1    9       0      0      0
  2    2     1   13       0      0      0

SSDs: 2 configured
  # indx unit iopath hspmsk errors recvrd unrcvd.
  0    0    0     01     03      0      0      0
  1    1    0     06     03      0      0      0
```

Example 3: The following example shows the `-p` option of the `ssd` command:

```
# /etc/ssd -p
Vhisps: 3 configured, 3 up, avail mask 07     (1 block = 64 words)
  # chan #transfers ..#.blocks.. blks/xfer .chan.time. ..time/xfer  .Xfer..Rate
  0    9          0            0       0.0         0ns         0ns 0.000000E+00
  1    1          4       524288  131072.0    269078us  67269635ns 9.976101E+08
  2    5          4       524288  131072.0    269061us  67265345ns 9.976737E+08

SSDs: 2 configured
  # unit iopath #transfers ..#.blocks.. blks/xfer #q’d maxq’d avg.q’d
  0    0     01          0            0       0.0    0      0   0.000
  1    0     06          2      1048576  524288.0    0      1   1.000
```

Example 4: The following example shows the `-q` option of the `ssd` command:

```
# /etc/ssd -q
SSDs: 2 configured
  # unit vhmask #q’d maxq’d  Qdonly Qdfirst  Qdlast Qdinbtw #cumq’d avg.q’d
  0    0     01    0      0       0       0       0       0       0   0.000
                                0.0%    0.0%    0.0%    0.0%

  1    0     06    0     17    6235       0    2768       0   31453   3.494
                               69.3%    0.0%   30.7%    0.0%
```

SEE ALSO

General UNICOS System Administration, Cray Research publication SG–2301

NAME

`ssddconf` – Controls the state of a model E SSD drive

SYNOPSIS

`ssddconf` [`-d` *device*] [*state*]

IMPLEMENTATION

Cray PVP systems with I/O subsystem model E

DESCRIPTION

The `ssddconf` command controls the state of a model E SSD drive. If no device is specified, all open devices are displayed. `ioctl`s are issued to both the logical and physical SSD drivers to mark the file systems that are affected by the SSD state change and mark the SSD driver tables that are necessary.

The `ssddconf` command accepts the following options:

`-d` *device*	Device I/O path (for example, `0132` or `0230.3`). If no device is specified, all open devices are shown.
state	One of the following:

`rw`	Sets the device to be operationally read/write.
`ronly`	Sets the physical device in read-only mode. Read requests are permitted, writes are returned with `errno` set to `EIO`, indicating an I/O error. If any mounted file systems are residing on that device, it marks that file system slice(s) not available for allocation.
`noall`	Sets a device to be operationally nonallocatable.
`up`	Sets the physical SSD device in the up state, and if any mounted file systems are residing on the SSD, marks that file system that its slice(s) are available for allocation.
`down`	Sets the physical device in the down state and terminates all queued I/O requests with errors. If there are any mounted file systems residing on that device, it marks that file system slice(s) not available for allocation.

The display has the following format:

```
command line: ssddconf

      SSD  devices
  iopath  type unit # open  state mode
 -------- ---- ---- ------ ------ ----
      01  SSD    0       2    up  rw
```

Where:

`iopath`	Device I/O path
`type`	Device type
`unit`	Device unit number
`state`	Device state (for example: up or down)
`mode`	Device mode: read/write (`rw`), read-only (`ro`), or nonallocatable (`na`)

SEE ALSO

`ioctl`(2)

NAME

`start_air` – Starts the automated incident reporting (AIR) system

SYNOPSIS

`/usr/air/bin/start_air`

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `start_air` shell script initiates the automated incident reporting (AIR) daemon (`aird`(8)) after checking for the existence of key AIR system components. By default, the `/etc/config/daemons` file initiates AIR by invoking this script.

The `start_air` shell script checks for the existence of the AIR home and log file directories and uses the `airexist`(8) command to check for a currently running `aird`(8) process. If one does not exist, `start_air` initiates an `aird`(8) process by using the default configuration and standard output files (`/usr/air/config_file` and `/usr/spool/air/logs/ttylog`, respectively).

SEE ALSO

`aird`(8), `airexist`(8)

UNICOS Resource Administration, Cray Research publication SG–2302

NAME

`startup` – Puts system in multiuser mode or turns on system accounting

SYNOPSIS

`/usr/lib/acct/startup`

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

This man page describes two `startup` shell scripts: `/etc/startup` puts the system in multiuser mode, and `/usr/lib/acct/startup` turns on system accounting.

The `/usr/lib/acct/startup` shell script turns on process accounting and, if directed, also turns on daemon accounting. `startup` checks the values of `NQS_START`, `TAPE_START` and `SOCKET_START` in the accounting configuration file `/etc/config/acct_config` to determine whether to turn on any daemon accounting. The `ACCTON` string, as defined in `/etc/config/acct_config`, is written to `/etc/wtmp` when startup is invoked.

Typically, `startup` is called from `/etc/rc` at each system boot.

NOTES

If Network Queuing System (NQS) accounting is enabled by `startup`, it also must be enabled by NQS by using the `qmgr set accounting on` command (see `qmgr`(8)).

If tape accounting is enabled by `startup`, you must use the `-c` option when starting the tape daemon, `tpdaemon`(8).

FILES

`/etc/config/acct_config`	Accounting configuration file
`/etc/wtmp`	Login/logoff summary
`/usr/adm/acct/day`	Directory that contains current accounting files

SEE ALSO

For the `/etc/startup` shell script:
`brc`(8), `init`(8), `killall`(8), `mount`(8), `slogdemon`(8)

For the `/usr/lib/acct/startup` shell script:
`acct`(8), `acctsh`(8), `brc`(8), `csa`(8), `rc`(8), `turnacct`(8), `turndacct`(8)

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

UNICOS Resource Administration, Cray Research publication SG–2302

NQE Administration, Cray Research publication SG–2150, for information about NQS accounting

NAME

`statd` – Provides the crash and recovery functions for NFS locking services

SYNOPSIS

`/etc/statd`

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `statd` daemon is an intermediate version of the status monitor. It interacts with `lockd`(8) to provide the crash and recovery functions for the locking services on the network file system (NFS).

NOTES

The crash of a site is detected only on its recovery.

FILES

`/usr/spool/statmon/sm`	Contains client names having current lock requests.
`/usr/spool/statmon/sm.bak`	Used in recovery of lock requests.
`/usr/spool/statmon/state`	Contains current state of `statd`.

SEE ALSO

`lockd`(8)

NAME

`staticrts` – Installs static routing information

SYNOPSIS

`/etc/staticrts` [`-n`] [`-f` *conffile*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `staticrts` command uses the `route`(8) command to install static routes defined in the `/etc/gated.conf` configuration file.

The `staticrts` command accepts the following options:

`-n`	Specifies that the `route` commands will be echoed, but not executed.
`-f` *conffile*	Specifies an alternative configuration file.

The routes listed in the configuration file are expected to be in the format of the dynamic routing daemon `gated`(8) (see the `gated-config`(5) man page for details). This is true regardless of whether the site actually chooses to run `gated`, to implement a single routing standard. `staticrts` searches the configuration file for any specifications of default routes, static network routes, or static host routes, and it issues the appropriate `route`(8) commands to install those routes in the kernel.

The `staticrts` command is intended primarily for those sites that choose not to run `gated`(8), and as an emergency backup in case the `gated`(8) program becomes unexecutable.

FILES

`/etc/gated.conf`	Default configuration file for routing information

SEE ALSO

`gated`(8), `netstart`(8), `route`(8)

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

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

NAME

`stdhosts` – Converts/prints Internet addresses from `hosts`(5) or `networks`(5) files

SYNOPSIS

`/etc/yp/stdhosts` [*file*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `stdhosts` command reads a file, converts the byte-ordered Internet addresses in the file to a standard format, and writes the converted file to standard output.

The `stdhosts` command accepts the following argument:

file Specifies the file to be read by `stdhosts`. If you do not specify a file, `stdhosts` reads from standard input. `stdhosts` expects the file to be in `hosts`(5) or `networks`(5) file format.

SEE ALSO

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

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

NAME

stor – Defines physical storage

SYNOPSIS

/etc/stor [-d *number*] *special_files*

IMPLEMENTATION

Cray PVP systems with I/O subsystem model E

CRAY J90 series

CRAY EL series

DESCRIPTION

The stor command sorts the special files by physical device numbers and writes, to standard output, information about starting and ending disk addresses and sizes.

The stor command accepts the following option and operand:

-d *number*	Prints those special files listed that match the specified physical device number.
special_files	Specifies one or more special files to process.

NOTES

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

Active Category	**Action**
system, secadm, sysadm	Allowed to specify any file.

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

EXAMPLES

The following command line displays information for disk device 0–101 only:

```
stor -d 0101 /dev/pdd/*
```

SEE ALSO

ddstat(8), mknod(8)

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

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

General UNICOS System Administration, Cray Research publication SG–2301

NAME

`suspend`, `resume` – Suspends or resumes a process or group of processes

SYNOPSIS

```
/etc/suspend [-p] pid ...
/etc/suspend [-g] pgrp ...
/etc/suspend [-j] jid ...

/etc/resume [-p] pid ...
/etc/resume [-g] pgrp ...
/etc/resume [-j] jid ...
```

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `suspend` procedure makes a process (or group of processes) ineligible to execute. The `resume` procedure restores a process (or group of processes) so it is eligible to execute.

The `suspend` and `resume` commands accept the following options and arguments:

`-p` *pid*	When invoked without arguments or with the `-p` option, affects one or more process IDs (*pid*).
`-g` *pgrp*	Affects process group IDs (*pgrp*).
`-j` *jid*	Affects job IDs (*jid*).

NOTES

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

Active Category	**Action**
`system`, `secadm`	Allowed to suspend or resume any process.
`sysadm`	Allowed to suspend or resume any process, 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 suspend or resume any process.

The `csh`(1) command has a built-in suspend command with slightly different characteristics. See `csh`(1) for more information.

In the standard shell, `suspend` is not a shell built-in, but an alias that the shell creates automatically. That is, `suspend` is actually an alias for `'kill -STOP $$'`.

SEE ALSO

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

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

UNICOS File Formats and Special Files Reference Manual, Cray Research publication SR–2014

NAME

`swapper` – Displays current system swapping activity

SYNOPSIS

`/etc/swapper [-f]`

IMPLEMENTATION

Cray PVP systems

DESCRIPTION

The `swapper` command displays current system swapping activity. It displays processes waiting to swap in and the priorities of in-memory processes that are candidates to swap out.

The swapper command prints the entries in the swap-in queue and then prints the processes that are available for swap-out if the swap-in queue is nonempty **and** one of the swapped processes is:

- Not suspended and has been out longer than the `Max swap-out time`

  or its UID is 0

  or its SUID is 0
- **and** its size is not greater than the total hog memory value

For any process in the swap-in queue that is greater than the total hog memory value, an asterisk is placed after the size field of that process.

The `swapper` command accepts the following option:

`-f` Force the printing of the swap-out candidates even if the swap-in queue is empty.

NOTES

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

Active Category	**Action**
`system`, `secadm`, `sysadm`	Retrieving process information is not subject to security label restrictions.

If the `PRIV_SU` configuration option is enabled, for the super user, retrieving process information is not subject to security label restrictions. Shell redirected I/O is not subject to security label restrictions.

EXAMPLES

```
Total swap space:  1748828      Available swap space:      1652136

Total swap-ins:        688      Total blocks swapped in:    297764
Total swap-outs:       782      Total blocks swapped out:   388976

Total hog memory:   224000      Available hog memory:       147116

Max swap-out time:       0      Max in-memory time:              0

Swap-in queue:

time    size    pri  shpri  nic    swpri     hog    pid   name
-----  -------  ---   ---   ---  ----------  ---   -----  --------
 7460       44   39     56   29   7459.6071  no     4252  quotamon
 7460       44   39     56   29   7459.6071  no    15598  quotamon
 7460       64   39     56   29   7459.4286  no     4243  chkpntd
 7460       64   39     56   29   7459.4286  no    15595  chkpntd
 7459       44   39     56   35   7458.6071  no     4172  quotamon
 7459       64   39     56   35   7458.4286  no     4169  chkpntd
 7460      228   56     56   29   7457.9643  yes    4267  a.out
 7460    13260   56     56   29   7341.6071  yes   15610  kiva
 7459    63360   39     56   35   6893.2857  yes    4195  fluent

Swap-out candidates:

time    size    pri  shpri  nic    swpri     hog    pid   name
-----  -------  ---   ---   ---  ----------  ---   -----  --------
low priority sleepers:
 1681      352   26     40   20     -3.1429  no     1944  ftp
 1681      152   26     46   20     -0.3214  no     2999  oper
  886       44   39     56   25      0.0000  no    10784  quotamon
  886       64   39     56   25      0.0000  no    10781  chkpntd
  701      160   39     40   20      0.0000  no    11268  csh
  186       60   26     40   20      0.0000  no    14876  telnetd
  400      292   26     56   24      0.0000  no    13013  sjc_11b
    1       64   30     52   20      0.0000  no    16330  sh
high priority sleepers:
  884    13836   20    487   25      0.0000  yes   10794  a.out
    1      436   20     71   20      0.0000  no    16333  cpp
 1205    11152   20    429   29      0.0000  yes    9353  Mc21.cray
   53     2012   20     60   24      0.0000  no    15879  EXE
```

```
runnable:
27590   11364   995   995    29  -25806.5357  yes    4761  a.out
12326   11364   994   994    29  -10543.5357  yes   47428  a.out
 8644    5820   987   987    27  -6911.0357  yes   60289  mndo91
 3380    5684   989   989    27  -1648.2500  yes   93721  new.out
 3104   13836   996   996    27  -1299.4643  yes   96757  tgie1
 2022    3828   987   987    27   -306.8214  yes    4981  kiva
 1681     140    57    57    20     -1.2500  no    85102  fta
    0     460    40    40    20      0.0000  no    16340  swapper
    1     164    68    68    20      0.0000  no    16337  quota
```

NAME

`sysctl` – Gets or sets kernel state

SYNOPSIS

`sysctl` [`-n`] *name* ...
`sysctl` [`-n`] `-w` *name=value* ...
`sysctl` [`-n`] `-a`
`sysctl` [`-n`] `-A`

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `sysctl` utility retrieves kernel state and allows processes with appropriate privilege to set kernel state. The state to be retrieved or set is described using a Management Information Base (MIB) style name, described as a dotted set of components. The `-a` flag can be used to list all the currently available string or integer values. The `-A` flag will list all the known MIB names including tables. Those with string or integer values will be printed as with the `-a` flag; for the table values, the name of the utility to retrieve them is given.

The `-n` flag specifies that the printing of the field name should be suppressed and that only its value should be output. This flag is useful for setting shell variables. For example, to save the maximum size of the *ipintrq* in variable *qlen*, use:

```
set qlen='sysctl -n net.inet.ip.qmaxlen'
```

If just a MIB style name is given, the corresponding value is retrieved. If a value is to be set, the `-w` flag must be specified and the MIB name followed by an equal sign and the new value to be used.

The information available from `sysctl` consists of integers, strings, and tables. The tabular information can only be retrieved by special purpose programs such as `netstat`, `arp`, and `spnet`. The string and integer information is summaried below. For a detailed description of these variables see `sysctl`(3C). The changeable column indicates whether a process with appropriate privilege can change the value.

Name	Type	Changeable
`net.unix.stream.sendspace`	`integer`	yes
`net.unix.stream.recvspace`	`integer`	yes
`net.unix.dgram.sendspace`	`integer`	yes
`net.unix.dgram.recvspace`	`integer`	yes
`net.inet.ip.forwarding`	`integer`	yes
`net.inet.ip.redirect`	`integer`	yes

Name	Type	Changeable
`net.inet.ip.qmaxlen`	`integer`	yes
`net.inet.ip.subnetsarelocal`	`integer`	yes
`net.inet.ip.mrtproto`	`integer`	no
`net.inet.ip.mrtdebug`	`integer`	yes
`net.inet.ip.dynamic_mtu`	`integer`	yes
`net.inet.ip.admin_override_mtu`	`integer`	yes
`net.inet.ip.ipmaxpkts`	`integer`	yes
`net.inet.ip.iploadleveling`	`integer`	yes
`net.inet.icmp.maskrepl`	`integer`	yes
`net.inet.tcp.printfs`	`integer`	yes
`net.inet.tcp.rexmtthresh`	`integer`	yes
`net.inet.tcp.defttl`	`integer`	yes
`net.inet.tcp.sendspace`	`integer`	yes
`net.inet.tcp.recvspace`	`integer`	yes
`net.inet.tcp.keepidle`	`integer`	yes
`net.inet.tcp.debx`	`integer`	no
`net.inet.tcp.ndebug`	`integer`	no
`net.inet.tcp.autowinshft`	`integer`	yes
`net.inet.udp.checksum`	`integer`	yes
`net.inet.udp.sendspace`	`integer`	yes
`net.inet.udp.recvspace`	`integer`	yes
`net.inet.udp.ttl`	`integer`	yes
`net.trace.tcpwaitq`	`integer`	no
`net.trace.udpwaitq`	`integer`	no
`net.trace.recvspace`	`integer`	yes
`net.link_layer.0.ifqmaxlen`	`integer`	yes
`net.link_layer.0.bufstatlen`	`integer`	no
`net.link_layer.0.bufstatsize`	`integer`	no
`net.sock.maxsock`	`integer`	yes
`net.sock.sbmax`	`integer`	yes
`net.sock.printdelay`	`integer`	yes
`user.cs_path`	`string`	no
`user.bc_base_max`	`integer`	no

Name	Type	Changeable
`user.bc_dim_max`	`integer`	no
`user.bc_scale_max`	`integer`	no
`user.bc_string_max`	`integer`	no
`user.coll_weights_max`	`integer`	no
`user.expr_nest_max`	`integer`	no
`user.line_max`	`integer`	no
`user.re_dup_max`	`integer`	no
`user.posix2_version`	`integer`	no
`user.posix2_c_bind`	`integer`	no
`user.posix2_c_dev`	`integer`	no
`user.posix2_char_term`	`integer`	no
`user.posix2_fort_dev`	`integer`	no
`user.posix2_fort_run`	`integer`	no
`user.posix2_localedef`	`integer`	no
`user.posix2_sw_dev`	`integer`	no
`user.posix2_upe`	`integer`	no
`user.stream_max`	`integer`	no
`user.tzname_max`	`integer`	no
`mbuf.nmbspace`	`integer`	no
`mbuf.mhbase`	`integer`	no
`mbuf.mdbase`	`integer`	no
`netsec.mls_enabled`	`integer`	no

EXAMPLES

To retrieve the maximum size of the *ipintrq*, use the following:

```
sysctl net.inet.ip.qmaxlen
```

To set the maximum size of the *ipintrq* to 100, use the following:

```
sysctl -w net.inet.ip.qmaxlen=100
```

FILES

`<sys/sysctl.h>`	Definitions for top level identifiers, second level kernel and hardware identifiers, and user level identifiers
`<sys/socket.h>`	Definitions for second level network identifiers
`<sys/un.h>`	Definitions for fourth level UNIX domain identifiers
`<net/if.h>`	Definitions for fourth level IF identifiers
`<netinet/in.h>`	Definitions for third level Internet identifiers and fourth level IP identifiers
`<netinet/icmp_var.h>`	Definitions for fourth level ICMP identifiers
`<netinet/igmp_var.h>`	Definitions for fourth level IGMP identifiers
`<netinet/tcp_var.h>`	Definitions for fourth level TCP identifiers
`<netinet/udp_var.h>`	Definitions for fourth level UDP identifiers
`<sys/tr_pcb.h>`	Definitions of fourth level TRACE domain identifiers

NOTES

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

Active Category	**Action**
`system`, `secadm`, `sysadm`	Allowed to use the [`-w`] option to set the kernal state.

If the `PRIV_SU` configuration option is enabled, the super user is allowed to use the [`-w`] option to set the kernal state.

SEE ALSO

`sysctl`(3C) in the *UNICOS Networking Facilities Administrator’s Guide*, Cray Research publication SG–2304

NAME

`syslogd` – Logs system messages

SYNOPSIS

`/etc/syslogd` [`-f` *configfile*] [`-m` *markinterval*] [`-p` *path*] [`-P` *port*] [`-d`]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `syslogd` command reads and logs system messages into a set of files described by the configuration file `/etc/syslog.conf`. Each message is one line of text. A message can contain a priority code, marked by a number in angle braces at the beginning of the line. Priorities are defined in the include file `sys/syslog.h`. `syslogd` reads from the named pipe `/dev/log`, from an Internet domain socket specified in `/etc/services` (see `services`(5)), and from the special device `/dev/klog` to read kernel messages (see `log`(4)).

The shell script `newsys`(8) starts `syslogd`; however, before it starts the daemon, `newsys` renames the log files so that they have a length of 0. This is because if the daemon simply started the files being renamed, they would eventually grow to fill the file systems on which they reside (`syslogd` never truncates the log files).

`newsys` also saves the 10 most recent copies of the files and deletes any older copies. After this is done for all log files, `newsys` starts the `syslogd` daemon.

The `syslogd` command accepts the following options:

`-f` *configfile*	Specifies an alternate configuration file.
`-m` *markinterval*	Selects the number of minutes between mark messages.
`-p` *path*	Specifies a named pipe other than `/dev/log`.
`-P` *port*	Specifies an alternate port. This is a handy feature if more than one `syslogd` is to run at the same time.
`-d`	Turns on debugging.

The `syslogd` command configures when it starts and whenever it receives a hang-up signal. Lines in the configuration file have a *selector* to determine the message priorities to which the line applies and an *action*. The *action* field is separated from the selector by one or more tabs.

Selectors are lists of priority specifiers, separated by semicolons. Each priority has a *facility* describing the part of the system that generated the message, a dot, and a *level* indicating the severity of the message. You may use symbolic names. Use an asterisk to select all facilities. All messages of the specified level or higher (greater severity) are selected. You may select more than one facility, using commas to separate them. Known facilities and levels recognized by `syslogd` are those listed in `syslog`(3C) without the prefix `LOG_`. The additional facility `mark` has a message at priority `LOG_INFO` sent to it every 20 minutes (this may be changed with the `-m` option). The `mark` facility is not enabled by a facility field containing an asterisk. The level `none` may be used to disable a particular facility. The second part of each line describes where the message is to be logged if this line is selected. There are four forms, as follows:

- A file name (beginning with a leading slash); the file is opened in append mode.
- A host name preceded by an at-sign (@); selected messages are forwarded to the `syslog` daemon on the specified host.
- A comma-separated list of users; selected messages are written to those users if they are logged in.
- An asterisk; selected messages are written to all logged-in users.

Blank lines and lines beginning with # are ignored.

`syslogd` creates the `/etc/syslog.pid` file, if possible, containing a single line with its process ID. You can use this to kill or reconfigure `syslogd.`

To bring `syslogd` down, send it a terminate signal (for example, `kill `cat /etc/syslog.pid``).

To change what is logged through `syslogd`, edit the configuration file (`/etc/syslog.conf`). The default configuration file is described in *General UNICOS System Administration*, Cray Research publication SG–2301.

NOTES

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

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

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

On a UNICOS system with the `FORCED_SOCKET` configuration option enabled, the named pipe interface, `/dev/log`, is not available. In this case, the `-p` option is forced to the TCP/IP default socket specified in the `/etc/services`. If the pipe interface is requested using the `-p` option, the request is ignored.

EXAMPLES

Example 1: Be careful when using wildcards when selecting facilities; the last level specified on the line is the one that is used. Therefore, the following line selects messages of the `err` level for the facility `kern`, not the `debug` level, because the wildcard applies to all facilities:

```
kern.debug;*.err
```

Example 2: The following line selects messages of the `debug` level for the facility `kern`:

```
*.err;kern.debug
```

Example 3: The following line selects all facilities at the `emerg` level and the `mail` and `daemon` facilities at the `crit` level:

```
*.emerg;mail,daemon.crit
```

Example 4: The following line sends all messages except mail messages to the selected file:

```
*.debug;mail.none
```

Example 5: The following configuration file logs all kernel messages and 20-minute marks onto the system console, all notice (or higher) level messages and all mail system messages, except debug messages, into the file `/usr/spool/adm/syslog`, and all critical messages into `/usr/adm/critical`. Kernel messages of error severity or higher are forwarded to ucbarpa. All users are informed of any emergency messages. Users `eric` and `kridle` are informed of any alert messages, and user `ralph` is informed of any alert message or any warning message (or higher) from the authorization system.

```
kern,mark.debug          /dev/console
*.notice;mail.info       /usr/spool/adm/syslog
*.crit                   /usr/adm/critical
kern.err                 @ucbarpa
*.emerg                  *
*.alert                  eric,kridle
*.alert;auth.warning     ralph
```

FILES

`/etc/syslog.conf`	Configuration file
`/etc/syslog.pid`	Process ID
`/dev/log`	Name of the named pipe
`/dev/klog`	Kernel log device

SEE ALSO

`newsys`(8)

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

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

`log`(4), `services`(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

`tcpstart` – Starts the TCP/IP networking software

SYNOPSIS

`/etc/tcpstart`

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `tcpstart` script is called from the `netstart`(8) command to initialize the TCP/IP networking software. `tcpstart` relies on underlying configuration files for system-specific initialization, and thus, it should never need modification.

The `tcpstart` command performs the following functions:

- Executes the local script `/etc/tcpstart.pre` to perform any local initializations configured by the system administrator.
- Sets kernel run-time networking variables by executing the `netvar`(8) command with the contents of the `/etc/config/netvar.conf` file as arguments.
- Creates the binary host file (see `mkbinhost`(8)).
- Sets the system host name, using the `hostname`(1) command to one of the following:
  - The output of the `/etc/config/makehostname` script if it exists and is executable
  - The contents of the `/etc/config/hostname.txt` file if it exists
  - The compiled-in kernel name, as reported by `uname`(1)
- Initializes the TCP/IP networking interfaces by executing the `initif`(8) script.
- Executes the local script `/etc/tcpstart.mid` to perform any local initializations, configured by the system administrator, that must be performed after the interfaces have been configured but before static routes have been installed in the kernel.
- Initializes routing by executing the `staticrts` script if the `gated`(8) daemon is not configured as part of system startup.
- Starts the TCP/IP networking daemons by calling the `sdaemon`(8) utility.
- Executes the local script `/etc/tcpstart.pst` to perform any local initializations configured by the system administrator.

FILES

`/etc/config/netvar.conf`	Start-up arguments to `netvar`
`/etc/config/makehostname`	Executable script to generate a host name
`/etc/config/hostname.txt`	File that contains system host name

SEE ALSO

`gated`(8), `initif`(8), `mkbinhost`(8), `netvar`(8), `sdaemon`(8), `staticrts`(8)

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

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

NAME

`telnetd` – Invokes the DARPA TELNET protocol server

SYNOPSIS

`/etc/telnetd` [`-debug`] [`-h`] [`-I`*initid*] [`-l`] [`-r` *lowpty*`-`*highpty*] [`-D options`] [`-D report`] [`-D exercise`] [`-D netdata`] [`-D ptydata`] [`-B`] [`-S` *tos*] [*port*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `telnetd` command is a server to the DARPA standard TELNET virtual terminal protocol. The TELNET server operates at the port indicated in the `telnet` service description; see `services`(5). Specify a port number on the command line to override this port number (for debugging, diagnostics, or `bftp`(1B) server mode).

The `telnetd` command operates by allocating a pseudo-terminal device (see `pty`(4)) for a client, then by creating a login process that has the slave side of the pseudo-terminal device as `stdin`, `stdout`, and `stderr`. `telnetd` manipulates the master side of the pseudo terminal, implementing the TELNET protocol and passing characters between the client and login process.

The `telnetd` command accepts the following options:

`-debug` Enables debugging on each socket created by `telnetd` (see `SO_DEBUG` in `socket`(2)).

`-h` Disables the printing of host-specific information before login has been completed.

`-I`*initid* Specifies the ID from `/etc/inittab` to use when `init` starts login sessions. The default ID is `fe`.

`-l` Specifies line mode. Tries to force clients to use line-at-a-time mode.

`-r` *lowpty*`-`*highpty*
Specifies an inclusive range of pseudo-terminal devices to use. If the system has `sysconf` variable `_SC_CRAY_NPTY` configured, the default pty search range is `0` to `_SC_CRAY_NPTY`; otherwise, the default range is 0 to 128. Either *lowpty* or *highpty* may be omitted to allow changing either end of the search range. If *lowpty* is omitted, the `-` character is still required so that `telnetd` can differentiate *highpty* from *lowpty*.

[`-D options`]
[`-D report`]
[`-D exercise`]
[`-D netdata`]

[-D ptydata]
Specifies diagnostics output from the telnet server. This is useful for debugging interoperability problems with telnet clients that do not have debugging capability of their own. A telnetd used as a telnet diagnostics server should be used only on a port that is different from the normal telnet port. One argument is required with the -D option; multiple arguments require separate -D options. For example, -D netdata ptydata will not have the desired effect; use -D netdata -D ptydata.

The options argument causes telnetd to include clear text of telnet options to be included in the output stream. This is similar to the options toggle available from telnet.

The report argument produces options as well as other information about the status of telnetd. Setting the options argument when the report argument has been set has no additional effect.

The exercise argument is ignored.

The netdata argument causes telnetd to echo data read from its network connection, similar to the netdata toggle available from telnet.

The ptydata argument causes telnetd to echo the data it writes to the pty that the user's process is reading from. This can be useful when used in conjunction with the netdata option to see what processing telnetd is performing on the incoming data stream, such as end-of-line processing.

-B Specifies bftp server mode. In this mode, telnetd causes login to start a bftp(1B) session rather than the user's normal shell. bftp daemon mode does not support normal logins, and it must be used on a port other than the normal telnet port.

-S *tos* Sets the IP Type-of-Service (TOS) option on the connection to the value *tos*, which may be a numeric TOS value or a symbolic TOS name found in the /etc/iptos file.

port Specifies an alternative port.

The UNICOS TCP/IP telnet daemon tries to use telnet-line-mode (see RFC 1116) to support line-at-a-time mode. If the client does not support line mode, the telnet daemon tries to use an alternative method of supporting line-at-a-time mode by proposing the use of telnet go-ahead (SGA) options.

The telnetd command supports the following TELNET options:

- Binary mode
- Status of options
- Suppress go-ahead
- Remote echoing
- Negotiate about window size
- telnet status
- Flow control

- Terminal type
- Terminal speed
- Options on the extended options list
- Automatic authentication (by using Kerberos)
- Data stream encryption (not supported outside of the USA and Canada)

It does not support timing mark. Currently, no options are defined on the extended options list.

The implementation of the TELNET options follow the TELNET specifications. For a detailed description of the options, see the TELNET specifications RFC 854 to 861, RFC 1073, RFC 1079, RFC 1080, RFC 1091, and RFC 1116.

Many `telnet` client implementations do not support line-at-a-time `telnet` sessions correctly.

NOTES

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

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

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

The `telnetd` command cannot be serviced directly; it must be run through `inetd`(8).

SEE ALSO

`inetd`(8)

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

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

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

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

NAME

`tftpd` – Server to the DARPA trivial file transfer protocol

SYNOPSIS

`/etc/tftpd` [`-d`] [*port*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `tftpd` program is a server to the DARPA Trivial File Transfer Protocol (TFTP). The TFTP server operates at the port indicated in the TFTP service description; see `services`(5). The `tftpd` program accepts the following option and operand:

`-d`	Enables debugging for each socket created by `tftpd` (see `SO_DEBUG` in `socket`(2)).
port	Overrides the port number in the TFTP service description (for debugging purposes). Instead, `tftp` uses the port number specified on the command line.

You do not need an account or password on the remote system to use `tftp`. Because of the lack of authentication information, `tftpd` allows only publicly readable files to be accessed by the default `get` and `put` commands. This extends the concept of ‘‘public’’ to include all users on all hosts that can be reached through the network; this may not be appropriate on all systems, and its implications should be considered before enabling `tftpd` service. However, `tftpd` also supports Kerberos-authenticated requests from hosts that provide a kerberized `tftp` client. For Kerberos-authenticated requests, `tftpd` provides access to all files readable by the authenticated network principal.

Because `tftpd` does not require an account name or password, inbound `tftpd` is a security problem for non-Kerberos-authenticated requests. To reduce this problem, the `/etc/tftpd.conf` file should be maintained by the system administrator. This file contains a list of directories that are accessible by `tftpd`. Each directory has to be specified on a separate line. The administrator must provide the absolute path names for these directories and their access permissions. Access permissions are as follows:

R	Read-only access by `tftpd`
W	Write-only access
RW or WR	Read/write access by `tftpd`

The `tftpd` access permissions differ from the directory’s normal access permissions, and the directory’s permissions override `tftpd`’s access. That is, if a directory cannot be read by others, putting that directory in this file with read permission does not automatically permit access.

BUGS

This server is known only to be self-consistent (that is, it operates with the user TFTP program, `tftp`(1B)). Because of the unreliability of the transport protocol (`udp`(4P)) and the scarcity of TFTP implementations, it is uncertain whether it really works.

The search permissions of the directories leading to the files accessed are not checked.

EXAMPLES

In the following example of the `/etc/tftpd.conf` file, `/garbage` is readable and writable, and `/tmp` is readable through `tftp`, provided that these directories have their world read/write and read permissions set, respectively.

By default, no directories are accessible by `tftpd`.

```
# This is the tftpd configuration file.  It contains
# the list of directories, and the type of access for
# these directories for tftpd.

# The format is as follows:
#       Mode    Directory
#       where Mode is either R (for only read access
#       via tftpd), W (only write) and RW or WR (for read
#       write access via tftpd).  Each directory has to
#       be specified on a separate line.
#
#
# NOTE:  tftpd is a big security risk and should not be used
# unless necessary.

      RW    /garbage
      R     /tmp
```

SEE ALSO

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

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

NAME

`tic` – Invokes the `terminfo` compiler

SYNOPSIS

`/usr/bin/tic` [`-v` [`n`]] [`-c`] *file*

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `tic` command translates a `terminfo`(5) file from the source format into the compiled format. The results are placed in the directory `/usr/lib/terminfo`. The compiled format is necessary for use with the library routines described in `curses`(3).

`-v` [`n`] (Verbose) Output to standard error trace information showing the progress of `tic`. The optional integer `n` is a number from 1 to 10, inclusive, indicating the desired level of detail of information. If `n` is omitted, the default level is 1. If `n` is specified and greater than 1, the level of detail is increased.

`-c` Check only *file* for errors. Errors in `use=` links are not detected.

file Contains one or more `terminfo`(5) terminal descriptions in source format (see `terminfo`(5)). Each description in the file describes the capabilities of a particular terminal. When a `use=`*entry-name* field is discovered in a terminal entry currently being compiled, `tic` reads in the binary from `/usr/lib/terminfo` to complete the entry. (Entries created from *file* will be used first. If the `TERMINFO` environment variable is set, that directory is searched instead of `/usr/lib/terminfo`.) `tic` duplicates the capabilities in `entry-name` for the current entry, with the exception of those capabilities that are defined explicitly in the current entry.

If the `TERMINFO` environment variable is set, the compiled results are placed there instead of `/usr/lib/terminfo`.

NOTES

The files in `/usr/lib/terminfo` are created by the `tic` administrator utility. This utility produces a word-oriented format. Although not installed, an `otic` is created during the build of the `libcurses` library. An administrator may use `otic` to build old (pre-UNICOS 7.0) binary terminal information files and place them in an alternate directory, such as `/usr/lib/oterminfo`. These files may then be accessed successfully by executable files by setting the `TERMINFO` shell variable (and exporting it, if necessary) to the alternate directory name. `setupterm` in `libcurses` will then read the terminal information for the directory specified by `TERMINFO`.

WARNINGS

Total compiled entries cannot exceed 4096 bytes. The name field cannot exceed 128 bytes.

When the `-c` option is used, duplicate terminal names will not be diagnosed; however, when `-c` is not used, they will be.

Use of the `-v` option should be done with great care. This option will cause `tic` to send terminal control (commmonly called escape) sequences to `stderr`. These escape sequences may directly affect the operation of the user's terminal.

ENVIRONMENT VARIABLES

`TERMINFO`	Identifies the path to an alternative terminal information (`TERMINFO`) directory. The `/usr/lib/terminfo` default directory file contains terminal definitions. You can create your own terminal definitions, compile them and store them in one of your own directories. For the system to recognize an alternative directory, you must set `TERMINFO` (and export it, if necessary). For example, you could set `TERMINFO=$HOME/terminfo`.

MESSAGES

Most diagnostic messages produced by `tic` during the compilation of the source file are preceded with the approximate line number and the name of the terminal currently being worked on.

`mkdir ... returned bad status`
: The named directory could not be created.

`File does not start with terminal names in column one`
: The first thing seen in the file, after comments, must be the list of terminal names.

`Token after a lseek not NAMES`
: Somehow the file being compiled changed during the compilation.

`Not enough memory for use_list element`
`Out of memory`
: Not enough free memory was available (`malloc`(3C) failed).

`Can't open ...`
: The named file could not be created.

`Error in writing ...`
: The named file could not be written to.

`Can't link ... to ...`
: A link failed.

`Error in re-reading compiled file ...`
: The compiled file could not be read back in.

`Premature EOF`
: The current entry ended prematurely.

`Backspaced off beginning of line`
: This error indicates something wrong happened within `tic`.

`Unknown Capability - "..."`
: The named invalid capability was found within the file.

`Wrong type used for capability "..."`
: For example, a string capability was given a numeric value.

`Unknown token type`
: Tokens must be followed by @ to cancel, `,` for Booleans, # for numbers, or `=` for strings.

`"...": bad term name`
`Line ...: Illegal terminal name - "..."`
`Terminal names must start with a letter or digit`
: The given name was invalid. Names must not contain white space or slashes, and must begin with a letter or digit.

`"...": terminal name too long.`
: An extremely long terminal name was found.

`"...": terminal name too short.`
: A one-letter name was found.

`"..." filename too long, truncating to "..."`
: The given name was truncated due to UNICOS system file name length limitations.

`"..." defined in more than one entry.`
: Entry being used is "...". An entry was found more than once.

`Terminal name "..." synonym for itself`
: A name was listed twice in the list of synonyms.

`At least one synonym should begin with a letter.`
: At least one of the names of the terminal should begin with a letter.

`Illegal character - "..."`
: The given invalid character was found in the input file.

`New-line in middle of terminal name`
: The trailing comma was probably left off the list of names.

`Missing comma`
: A comma was missing.

`Missing numeric value`
: The number was missing after a numeric capability.

`NULL string value`
: The proper way to say that a string capability does not exist is to cancel it.

`Very long string found.  Missing comma?`
: Self-explanatory.

`Unknown option. Usage is:`
: An option that was not valid was entered.

`Too many file names.  Usage is:`
: Self-explanatory.

`"..." nonexistent or permission denied`
: The given directory could not be written into.

`"..." is not a directory`
: Self-explanatory.

`"...": Permission denied`
: Access denied.

`"...": Not a directory`
: `tic` wanted to use the given name as a directory, but it already exists as a file.

`SYSTEM ERROR!! Fork failed!!!`
: A `fork`(2) failed.

`Error in following up use-links.`
: Either there is a loop in the links or they reference nonexistent terminals. The following is a list of the entries involved.

  A `terminfo`(5) entry with a `use=`*name* capability either referenced a nonexistent terminal called *name*, or *name* somehow referred back to the given entry.

BUGS

To allow existing executables from the previous release of the UNICOS system to continue to run with the compiled `terminfo` entries created by the new `terminfo` compiler, cancelled capabilities will not be marked as cancelled within the `terminfo` binary unless the entry name has a `+` within it. (Such terminal names are only used for inclusion within other entries via a `use=` entry. Such names would not be used for real terminal names.)

Example:

```
4415+nl, kf1@, kf2@, ...

4415+base, kf1=\EOc, kf2=\EOd, ...

4415-nl|4415 terminal without keys,
   use=4415+nl, use=4415+base,
```

The above example works as expected; the definitions for the keys do not show up in the `4415-nl` entry. However, if the entry `4415+nl` did not have a plus sign within its name, the cancellations would not be marked within the compiled file and the definitions for the function keys would not be cancelled within `4415-nl`.

FILES

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

SEE ALSO

`infocmp`(8)

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

`curses`(3) (available only online)

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

NAME

`tpapm` – Tape daemon autoloader premount program

SYNOPSIS

`/etc/tpapm` [`-g` *device-group-name*] [`-l` *label-type*] [`-m` *message-file*] [`-r` *ring-option*] *vsn*

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `tpapm` command requests that the tape daemon mount a volume (*vsn*) on any available drive that may be serviced by an autoloader. This function is useful when you must premount volumes required by some program (for example, a backup) that is performing a multivolume operation, and the required volumes reside in the autoloader storage unit (silo).

The `tpapm` command accepts the following options and operand:

`-g` *device-group-name*	Specifies the name of the group to which the volume should be mounted. If you do not specify the device group name, it defaults to an installation-specified name. The device group name (*dgn*) field of `tpstat`(1) shows the allowable parameters.
`-l` *label-type*	Defaults to an installation-specified label type. The label type may be one of the following:

`al`	ANSI label.
`blp`	Bypass label processing.
`nl`	Not labeled.
`sl`	IBM standard label.
`st`	Single-tape mark format; single tape mark terminates reel.

Note: Special permission is needed for `blp` and `nl` labels. See your system administrator.

`-m` *message-file*	Specifies a file to which informative messages from the tape subsystem are written. The default message file is `/dev/null`.
`-r` *ring-option*	Specifies whether the write protect ring is in or out. The allowable value for *ring-option* is either `in` or `out`. If you do not specify `-r`, an installation-defined default is used.
vsn	Specifies the volume serial number (VSN) to mount on any available device.

NOTES

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

Privilege Text	Action
SMACDAC	Allowed to use this command if you have an active secadm category.
SDAC	Allowed to use this command if you have an active sysadm category.

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

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

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

The tpapm command is not functional if the Cray/REELlibrarian (CRL) is in the mandatory state. For information on displaying the state CRL is in, see tpset(8).

SEE ALSO

tpmql(8)

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

General UNICOS System Administration, Cray Research publication SG–2301

Tape Subsystem Administration, Cray Research publication SG–2307

NAME

`tpbmx` – Displays operator information about tape devices

SYNOPSIS

```
/etc/tpbmx [-d]
/etc/tpbmx device_name
```

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `tpbmx` command displays tape device information that may be useful to operators and system administrators. This information comes primarily from the `bmxtab` structure for each device in the kernel (the definition of `bmxtab` is in `/usr/include/sys/bmx.h`).

The `tpbmx` command accepts the following option and operand:

`-d` Displays the `bmxtab` structure for all devices.

device_name Specifies the device for which to display the `bmxtab` structure.

If you do not specify an option or operand, `tpbmx` displays a subset of the fields in the `bmxtab` structure.

The following information is provided on the status output:

device The device name for each device.

pm Path mask. The path mask is a hexadecimal digit that specifies active paths. Bit 0 is the rightmost bit; bit 3 is the leftmost bit. Bit 0 represents path 1.

st Device status.

`dn` Specifies down status.

`up` Specifies up status.

flags Flags used by the tape driver.

Symbol	Value	Description
`TPD_OPEN`	1	Device open for I/O
`TPD_ABN`	2	Device abnormal status
`TPD_OIP`	4	Device open in progress
`TPD_NOREAD`	010	No read ahead
`TPD_BKSPC`	020	Back up to a file mark
`TPD_RDONLY`	040	Device read only
`TPD_USRSAVED`	0100	Saved user's I/O pointers
`TPD_UERR`	0200	User error flag

Symbol	Value	Description
`TPD_DISCARD`	0400	Discard IOP replies
`TPD_PCLOSE`	0100	Pending close
`TPD_IORDY`	02000	Device ready for I/O
`TPD_HOLD`	04000	Close with hold
`TPD_USRIO`	010000	User first I/O complete
`TPD_DEMON`	020000	Under daemon control
`TPD_DIAG`	040000	Under diagnostic control
`TPD_SLEEP`	0100000	Process waiting on interrupt
`TPD_WIOQUIET`	0200000	Wait until I/O quiet
`TPD_UEOV`	0400000	Process user EOV
`TPD_CHKD`	01000000	Ordinal checked
`TPD_CLEAR`	02000000	Cleared by operator
`TPD_CIP`	04000000	Close in progress
`TPD_DEMOPEN`	010000000	Opened for daemon driver
`TPD_BMXOPEN`	020000000	Opened for BMX driver
`TPD_UNBUF`	040000000	Basic unbuffered I/O
`TPD_LIST`	0100000000	User-supplied list I/O
`TPD_MLIST`	0200000000	User-supplied multilist
`TPD_HLD`	0400000000	`FDF_HLD` set
`TPD_GBLK`	01000000000	`geteblk` in progress
`TPD_WAITBF`	02000000000	Waiting to get a `BF`
`TPD_WSIGDEM`	04000000000	Waiting to signal daemon
`TPD_ONEMORE`	010000000000	Write one block at EOT
`TPD_USYSBF`	020000000000	Use system buffer
`TPD_ACKERR`	040000000000	User `ack` error
`TPD_OPEN_DELAY`	0100000000000	User delayed in open
`TPD_WRITE`	0200000000000	Writing to tape
`TPD_RECHECK`	0400000000000	Recheck response
`TPD_REISSUE`	01000000000000	Reissue the command
`TPD_INTERRUPT`	02000000000000	In interrupt routine
`TPD_RESTART`	04000000000000	Waiting to restart I/O
`TPD_ERPA`	010000000000000	Call `ERPA` routine
`TPD_MODELE`	020000000000000	This is IOS model E
`TPD_LOCKP`	040000000000000	Process locked in memory
`TPD_INITIALIZED`	0100000000000000	TPD table initialized
`TPD_FLUSHBUF`	0200000000000000	Flush write buffer
`TPD_ROPE`	0400000000000000	A reopen request
`TPD_EOD`	01000000000000000	At end-of-data
`TPD_RECOVERY`	02000000000000000	In error recovery
`TPD_UEOVAPP`	04000000000000000	User EOV append read

Symbol	Value	Description
TPD_PACKET	010000000000000000	Use packet interface
TPD_WAITRCV	020000000000000000	Waiting for recovery
TPD_RCVDONE	040000000000000000	Recovery routine done
TPD_DENSITY	0100000000000000000	Density is set
TPD_TEST	0200000000000000000	A flag for testing
TPD_LOCKSETFL	0400000000000000000	Lock the BXC_SETFL ioctl
TPD_WTM_UEOV	01000000000000000000	Tape mark at UEOV time
TPD_TIMEDOUT	02000000000000000000	Request timed out
TPD_ARMED	04000000000000000000	Device is armed
TPD_RCVIO	010000000000000000000	Do recovery I/O
TPD_NOCHECK	020000000000000000000	Do not do any checking
TPD_CLOSEUNLD	040000000000000000000	Close unload in progress
TPD_PRIMED	0100000000000000000000	Device primed
TPD_READOPPOSIT	0200000000000000000000	Do read opposite

lreq Last request code.

For IBM compatible devices, the values for this flag are one or more of the following:

Symbol	Value	Description
echo	0001	Echo
lmcr	0202	Load adapter microcode
rmcr	0203	Read adapter microcode
strd	0004	Start driver
stpd	0005	Stop driver
abnk	0006	Add bank
dbnk	0007	Delete bank
achn	0011	Add channel
dchn	0012	Delete channel
cchn	0013	Change channel
acu	0014	Add control unit
dcu	0015	Delete control unit
ccu	0016	Change control unit
adev	0017	Add device
ddev	0020	Delete device
cdev	0021	Change device
cfcu	0022	Configure channel up
cfcd	0023	Configure channel down
chst	0024	Report channel state

Symbol	Value	Description
`dvst`	0025	Report device state
`armd`	0026	Arm device
`carm`	0027	Cancel arm
`opdv`	0030	Open device
`cldv`	0031	Close device
`cmdl`	0232	Command list
`selr`	0033	Selective reset
`hio`	0034	Halt I/O
`sysr`	0035	System reset
`drst`	0036	Report driver statistics
`bkst`	0037	Report bank statistics
`chst`	0040	Report channel statistics
`cust`	0041	Report control unit statistics
`dvst`	0042	Report device statistics
`opch`	0043	Open channel
`clch`	0044	Close channel
`wchb`	0245	Write channel buffer
`rchb`	0246	Read channel buffer
`chbp`	0047	Set channel buffer pointer
`wrqi`	0050	Enable and wait for request input
`drqi`	0051	Disable request in interrupts
`adpf`	0052	Issue adapter function
`rdrt`	0053	Read driver tables
`rbkt`	0054	Read bank tables
`rcht`	0055	Read channel tables
`rcut`	0056	Read control unit tables
`rdvt`	0057	Read device tables
`rtrb`	0260	Read trace buffer
`pdsc`	0061	Preset driver simulation code
`schs`	0062	Start channel simulator
`stcs`	0063	Stop channel simulator

For block multiplexer (mux) devices connected to an ESCON channel, the following requests are added to the above list:

Symbol	Value	Description
`cuup`	0064	Configure control unit up
`cudn`	0065	Configure control unit down

Symbol	Value	Description
`enas`	0066	Enable asynchronous responses
`asre`	0067	Asynchronous response

For ER90 devices, the values for this flag are one or more of the following:

Symbol	Value	Description
`lmcr`	0202	Load microcode
`rmcr`	0203	Read microcode
`strd`	0004	Start driver
`stpd`	0005	Stop driver
`abnk`	0006	Add bank
`achn`	0011	Add channel
`adsl`	0014	Add slave
`adev`	0017	Add device
`cfcu`	0022	Configure channel up
`cfcd`	0023	Configure channel down
`opdv`	0026	Open device
`enas`	0027	Enable async responses
`cldv`	0030	Close device
`cmdl`	0231	Command list
`hio`	0032	Halt I/O
`selr`	0033	Selective reset
`mstr`	0034	Master reset
`asyn`	0035	Asynchronous response

lrep	Last reply code. The values for this flag are the same as those for *lreq*.
resp	Response code. The values for this flag are the same as those for *lreq*.
bblk	Number of blocks in buffer.
bsec	Number of sectors in buffer. A sector consists of 4096 bytes.
or	Number of outstanding requests.
oblk	Number of outstanding blocks to transfer.
osec	Number of outstanding sectors to transfer.
fnum	File number. Number of tape marks encountered in the forward direction.

blks	Block number. This number reflects the direction the tape is moving, and the number of blocks of the tape that have moved in that direction. The block number increases when the tape is moving forward and decreases when it is moving backward.
pid	Process ID of the process that opened a file on this tape unit. If the process ID is 0, no process has opened a file on this tape.

NOTES

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

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

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

EXAMPLES

Example 1: This example shows system output. The tape daemon uses the special device name, `tpdemreq`, to make nondevice-specific requests to the driver. `tpdemreq` always is included. Device `170` has two available paths.

```
device    pm st flags                 lreq lrep resp bblk bsec or oblk fnum blks pid
tpdemreq  0 dn 0000000000000000000000 cfcu cfcu 0000 0000 0000 00 0000 0000 0000 0      0
200       1 dn 0000000000000000000000 0000 0000 0000 0000 0000 00 0000 0000 0000 0      0
201       1 dn 0000000000000000000000 0000 0000 0000 0000 0000 00 0000 0000 0000 0      0
202       1 dn 0000000000000000000000 0000 0000 0000 0000 0000 00 0000 0000 0000 0      0
203       1 dn 0000000000000000000000 0000 0000 0000 0000 0000 00 0000 0000 0000 0      0
170       3 up 0000100100200210012001 rwnd rwnd 0000 0000 0000 00 0000 0000 0000 0  71328
```

SEE ALSO

General UNICOS System Administration, Cray Research publication SG–2301

Tape Subsystem Administration, Cray Research publication SG–2307

NAME

`tpclr` – Clears the tape drive

SYNOPSIS

`/etc/tpclr [-r]` *device_name*

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `tpclr` command clears the last pending request on the data path to a tape drive. All tables and data associated with that device are cleared if possible.

The `tpclr` command accepts the following option and operand:

`-r`	Issues a slave reset to the ER90 device. A slave reset causes the ER90 device to reset itself to the initial start-up state. All information in the slave is lost. This option is necessary only if the device is hung. A slave reset clears out the ER90 traces. If possible, you should save the device traces before issuing this type of clear. By default, a logical reset is sent to the device. A logical reset clears out only the slave commands. It neither resets the device to the start-up state nor clears the device traces. If the device is not an ER90 device, this option is ignored.
device_name	Specifies name of device to be cleared.

NOTES

You should use the `tpclr` command only as a last resort. Use the `tpfrls`(8) command and other commands first.

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

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

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

EXIT STATUS

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

FILES

`/usr/include/taperr.h`	Tape daemon error codes

SEE ALSO

`tpfrls`(8)

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

General UNICOS System Administration, Cray Research publication SG–2301

Tape Subsystem Administration, Cray Research publication SG–2307

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

NAME

`tpconf` – Verifies new format tape daemon configuration file and converts it to binary format

SYNOPSIS

`/usr/lib/tp/tpconf` [`-i` *config-file*] [`-o` *bin-config-file*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `tpconf` command verifies the new format tape daemon configuration file and converts it to binary format for the tape daemon to process when it is started.

The `tpconf` command accepts the following:

`-i` *config-file*	Specifies the configuration file. If you omit this option, the standard input will be read.
`-o` *bin-config-file*	Specifies the binary format configuration file. If you omit this option, the output will be sent to the standard output.

For a system with an IOS model E, `tpconf` adds diagnostic devices. The user must not specify these devices. The diagnostic devices are as follows:

- One per IOP. The name is `IOP`*nn*; *nn* is the IOP number.
- One per channel. The name is `CHAN`*ipcc*; *i* is the cluster number, *p* is the IOP number, and *cc* is the channel number (in octal).

This increases the number of tape devices in the system. The `TAPE_MAX_DEV` parameter in the system boot parameter file should be the sum of the real devices, the number of tape IOPs, and the number of tape channels.

NOTES

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

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

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

EXIT STATUS

If no error occurs, the `tpconf` command exits with a return value of 0. If an error occurs, `tpconf` exits with a return value of nonzero.

SEE ALSO

`tpdaemon`(8)

General UNICOS System Administration, Cray Research publication SG–2301

Tape Subsystem Administration, Cray Research publication SG–2307

NAME

`tpconfig` – Configures tape devices up and down

SYNOPSIS

`/etc/tpconfig -m` *media_loader status*
`/etc/tpconfig -m` *media_loader device_name*
`/etc/tpconfig -m` *media_loader device_name*`:`*device_name*`:`...
`/etc/tpconfig -m` *media_loader device_name*`-`*device_name*
`/etc/tpconfig -g` *device_group_name device_name*
`/etc/tpconfig -g` *device_group_name device_name*`:`*device_name*`:`...
`/etc/tpconfig -g` *device_group_name device_name*`-`*device_name*
`/etc/tpconfig -b` *bank* `up|down`
`/etc/tpconfig [-n]` *device_name* `up|down`
`/etc/tpconfig [-n]` *device_name*`:`*device_name*`:`... `up|down`
`/etc/tpconfig [-n]` *device_name*`-`*device_name* `up|down`

For systems with an IOS model E:
`/etc/tpconfig -c` *ipc* `up|down`
`/etc/tpconfig -u` *ipc*/*control_unit* `up|down`
`/etc/tpconfig [-n] -d` *ipc*/*control_unit*/*device* `up|down`

For systems without an ESCON channel:
`/etc/tpconfig -c` *rrnncc* `up|down`
`/etc/tpconfig -u` *rrnncc*/*control_unit* `up|down`
`/etc/tpconfig [-n] -d` *rrnncc*/*control_unit/device* `up|down`

For control units connected via an ESCON channel:
`/etc/tpconfig -u` *rrnncc*/*link_address* `up|down`
`/etc/tpconfig [-n] -d` *rrnncc*/*link_address*/*device* `up|down`

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `tpconfig` command configures tape devices up and down, changes the status of the associated media loaders, assigns a media loader to a device, and reassigns a device group to a device. When options are not used, the *device_name* operand is required. Control unit numbers are in hexadecimal; all other numbers used as arguments and operands are octal.

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

`-m` *media_loader*
Specifies one of the valid loaders defined in the system.

status The *status* operand may have the following values:

`up` Configures the media loader up.

`down` Configures the media loader down.

`attd` Configures the media loader to attended mode.

`auto` Configures the Storage Technology Corporation (StorageTeK) autoloader to unattended mode or the IBM autoloader to auto mode.

device_name Specifies the device name of the device to be processed by `tpconfig`.

device_name`:`*device_name*`:`. . .
Specifies a list of device names separated by colons (:). Each device name identifies a device to be processed by `tpconfig`.

device_name`-`*device_name*
Specifies a range of devices to be processed by `tpconfig`. To indicate the range, specify the first device name and the last device name, separated by a dash (–). The order of the devices is determined by the order in which they appear in the output of the `tpstat`(1) command. If the first device does not appear before the last device, an error is returned.

`-g` *device_group_name*
Specifies a device group name to be reassigned to a device. The device group name must be valid in the `text_tapeconfig`(5) file. A device assigned to group `CART` could be reassigned to group `OPER`.

You can use this option to move a device or devices into a device group that is used by isolated users, such as operations. For example, you could set aside devices for system dumps. You could also put a tape drive into a special device group and reserve it for `tplabel`(8) tapes so that they do no have to compete for the device with general users.

`-b` *bank* Specifies a bank to be configured. A *bank* is a set of devices that have the same data paths. You cannot use this option with any other options.

`-n` Specifies that the following devices will not be unloaded (no unload). You can use this option to disable the automatic unloading of a volume when the user releases the tape for tapes that will be used repeatedly. When the device is configured down, this option returns an error. You can force a device to be unloaded by using the `tpu`(8) command.

If a tape is already mounted on the device, you can specify the `-n` option to keep the tape on the device. This option applies if the `status` option is either `up` or `down`.

If a tape is already mounted on the device, and you specify `status` as either `up` or `down` but omit the -n option, the tape is unloaded.

`-c` *rrnncc* Specifies a 2-digit ring number, a 2-digit node number, and a 2-digit channel number to be configured. You can use this specification only on GigaRing based systems.

`-u` *rrnncc*/*control unit*

Configures a control unit. Because a system may have several channels, and control unit numbers may not be unique across channels, you must specify a channel number with the control unit number. The following components compose the control unit configuration: a 2-digit ring number, a 2-digit node number, a 2-digit channel number, a slash (/), and a control unit number. You can use this specification only on GigaRing based systems.

`-d` *rrnncc*/*control_unit*/*device*

Specifies a device to be configured. The following components determine the data path to a tape device: a 2-digit ring number, a 2-digit node number, a 2-digit channel number, a slash (/), a control unit number, a slash (/), and a tape device number. You can use this specification only on GigaRing based systems.

`-d` *rrnncc*/*link_address*/*device*

Specifies device to be configured. For control units connected by an ESCON channel, the *control_unit* field is replaced by the *link_address*. You can use this specification only on GigaRing based systems.

NOTES

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

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

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

When you use `tpconfig`, replace any reference to a control unit that is connected by an ESCON channel with a link address. The link address defines the port on which a control unit is attached to the ESCON director. If there is no ESCON director in the path being described, the link address field must be set to 0. If an ESCON director exists in the path being described, the link address must reflect the port on which the control unit is attached. Values in the range 0. to .255 are valid.

EXIT STATUS

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

The device `tpconfig` operation is an asynchronous process and as such the initial exit status is zero. If an error occurs during the configuration process, the status is displayed on `/usr/spool/tape/daemon.stderr` and the device status reflects the problem accordingly.

EXAMPLES

The following examples show possible `tpconfig` command lines and tape configurations:

Example Command	**Action**
`tpconfig -c 20 up`	Configures channel 20 up.
`tpconfig -c 20 -i 1 down`	Configures channel 20 on IOS 1 down.
`tpconfig -u 20/1 down`	Configures control unit 1 on channel 20 down.
`tpconfig -u 20/1 -i 0 up`	Configures control unit 1 on channel 20 on IOS 0 up.
`tpconfig -d 20/1/3 down`	Configures a device on channel 20, control unit 1, device ID 3 down.
`tpconfig -b 1 up`	Configures bank number 1 up.
`tpconfig tape03 up`	Configures device `tape03` up.
`tpconfig tape03:tape01 down`	Configures devices `tape01` and `tape03` down.
`tpconfig tape01-cart00 up`	Configures devices from `tape01` to `cart00` up.
`tpconfig -n tape200 up`	Configures device `tape200` up and sets `no unload`.
`tpconfig -m stksun up`	Configures media loader `stksun` up.
`tpconfig -m stkvm 302`	Configures device 302 to be assigned to media loader `stkvm`.
`tpconfig -c 0132 up`	Configures channel 32 in IOS cluster 0, IOP number 1 up.
`tpconfig -u 0132/A up`	Configures control unit A of channel 32 in IOS cluster 0, IOP number 1 up.
`tpconfig -d 0132/A/2 up`	Configures tape unit 2 of control unit A up. Control unit A is connected to channel 32 in IOS cluster 0, IOP number 1.
`tpconfig -g OPER 301`	Assigns device group `OPER` to device 301.

FILES

`/dev/tape/`*device_name*	Tape device node
`/etc/config/text_tapeconfig`	Tape subsystem configuration file
`/usr/include/taperr.h`	Tape daemon error codes

SEE ALSO

`tpbmx`(8), `tpdaemon`(8), `tpdev`(8), `tpdstop`(8), `tpfrls`(8), `tplabel`(8), `tpgstat`(8), `tpu`(8)

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

`text_tapeconfig`(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

Tape Subsystem Administration, Cray Research publication SG–2307

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

NAME

`tpcore` – Initiates the tape subsystem monitor

SYNOPSIS

`/etc/tpcore` [`-a`] [`-A`] [`-b` *beep_rate*] [`-c` *core_file*] [`-d` *device_name*] [`-g`] [`-G`] [`-J`] [`-L`] [`-M`] [`-o` *obj_file*] [`-O`] [`-p` *pid*] [`-P`] [`-r` *refresh_rate*] [`-S` *screens*] [`-T` *tail_count*] [`-v`] [`-V` *vsn_count*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `tpcore` command initiates the tape subsystem monitor. Administrators use the monitor interactively to perform several functions on a running tape subsystem. These functions include monitoring the tape daemon and issuing warning messages when potentially dangerous conditions occur.

The `tpcore` command builds a buffer display of 10 screens of data. In order for the command to display this data, the `COLUMNS` and `LINES` environment variables must be set. By default, `tpcore` displays the online `UP` devices and any mount messages.

You may scroll through the display with the `+` (forward) and `-` (backward) commands. You may also control the tape daemon and send the operator `wto` messages from the command line.

The `tpcore` command accepts the following options:

`-a`	Disables the volume serial number (VSN) list display for active users. By default, all VSNs are listed on the assigned drive status.
`-A`	Displays only assigned drives.
`-b` *beep_rate*	Sets the interval between beeps. `tpcore` issues these beeps when there are outstanding mount messages. To disable beeping, set the `-b` option to 0.
`-c` *core_file*	Specifies the core file to use for monitoring. The default is `/proc/`*pid* where *pid* is determined from kernel information.
`-d` *device_name*	Displays all the information relating to the specified device.
`-g`	Skips the `tpgstat`(8) display. The default is to display a formatted `tpgstat`(8) display.
`-G`	Shows the number of users currently awaiting device allocation instead of the `tpgstat`(8) display.
`-J`	Displays the tape daemon child processes. This information is useful in determining the activity associated with a device while the user is suspended for some tape daemon service.
`-L`	Displays loader status by using an internal `tpmls`(8) command.

`-M`	Displays the internal tape daemon message queue. This queue is typically a superset of the messages that an operator can view through the `oper`(8) command.
`-o` *obj_file*	Specifies the object file for symbol table lookup. The search path default is `./tpdaemon`, then `TPDAEMON=obj_file` from the environment, followed by `/usr/lib/tp/tpdaemon`.
`-O`	Formats data dumped in octal.
`-p` *pid*	Specifies the PID to use for `/proc/pid` as the core file. The PID is determined automatically from an internal `ps`(1) command if you omit the `-p` option.
`-P`	Sets the display to page align mode. Major sections of the display are aligned on separate pages or screens of the interactive display.
`-r` *refresh_rate*	Sets the screen update rate to *refresh_rate*.
`-S` *screens*	Sets the number of screens of data to *screens*. The default number of screens is 10. Reducing this number may impact the information that is displayed.
`-T` *tail_count*	Generates a modified display of the last few lines in the `/usr/spool/tape/trace` file. The date and real time stamp are removed from the display and the data is formatted to fit the window width. The default is to tail 4 lines of `daemon.stderr`. To disable the `-T` option, set it to 0.
`-v`	Sets verbose mode.
`-V` *vsn_count*	Limits the number of VSNs displayed to *vsn_count*. This option is useful when memory is corrupted and the file information table overlaid.

The bottom line of the window is a command line, from which you may enter the following commands to control the interactive display and the tape subsystem.

Command **Description**

`add` *command* Adds a user-defined display to the output generated by the `tpcore` command.

The output of this shell command is displayed as the last data in the last page or screen. The command must exit; otherwise, the window hangs on its completion as shown in the following example:

```
add tail,16 a.file
```

Adding this `tail` command would render the monitor useless.

An `add` command that is issued while a previous command is active replaces the old command with the new command. For example, an administrator issues the following to watch device level activity through a `tpcore` display.

```
add tpbmx | sed "/0000000000000000000000/d"
```

`chopt  +|- a|A|g|J|L|M|P`
: Toggles the command line options. A plus (`+`) turns the option on and a minus (`-`) turns it off.

  For example, you can turn off the display on tape daemon child processes by using the `-J` option after `tpcore` has been started with the `-J` option enabled. Invalid toggle requests are ignored.

`del`
: Deletes the command specified with the `add` command.

`ref` *rate*
: Alters the refresh rate to *rate*. See the `-r` option.

`rep nn str`
: Enables you to respond to an outstanding operator message.

`tail`[,*num_lines*]*bmx_file*
: Adds the specified number lines of the `/usr/spool/tape/trace/`*bmx_file* file to the window. See the `-T` option.

  num_lines indicates the number of lines. If *num_lines* is less than 4 or more than 40, 4 lines are added. To delete this display, issue the `tail` command without any file name.

  The trace file date is only updated when the child process exits or sleeps during tape mark processing; so the data displayed may not accurately reflect the state of the currently executing child process.

  The data is used to detect stalled child processes and child exits abnormally. For example, when a child process begins executing, the trace entry provides the following kind of information:

```
    ...execpgm child for 120 started: jid =...
```

  If the trace file fails to update, it indicates that the child is waiting for some event to occur. This event can be a normal one such as a tape mount or an abnormal one such as looping.

`tpclr` *opts*
: Executes a `tpclr`(8) command through the `system`(3C) interface. For a description of *opts*, see the options and operands on the `tpclr`(8) man page.

`tpconfig` *opts*
: Executes a `tpconfig`(8) command through the `system`(3C) interface. For a description of *opts*, see the options and operands on the `tpconfig`(8) man page.

`tpfrls` *opts*
: Executes a `tpfrls`(8) command through the `system`(3C) interface. For a description of *opts*, see the options and operands on the `tpfrls`(8) man page.

`tpset` *opts*
: Executes a `tpset`(8) command through the `system`(3C) interface. For a description of *opts*, see the options and operands on the `tpset`(8) man page.

`tpu` *opts*
: Executes a `tpu`(8) command through the `system`(3C) interface. For a description of *opts*, see the options and operands on the `tpu`(8) man page.

`wto` *message*	Executes a background `/usr/bin/msgr` with *message* as the text, but does not wait for a response from the operator.
`^L`	Causes the screen to be redrawn.
`^U`	Erases current input line.
`^H`	Backspaces current command line position.
`+`	Advances to the next screen in the buffer.
`-`	Backs up one screen.
`!`(*command*)	Executes an immediate command. May be used to exit to a C shell as shown in the following `csh`(1) command example: `!csh`

NOTES

If this command is installed on a UNICOS MLS system with the default privilege assignment list (PAL), you must have an active `secadm`, `sysadm`, or `sysops` category to use this command. To override the MAC and DAC restrictions, you must have the `SMACDAC` privilege text (if you are a security administrator) or the SDAC system administrator). On a UNICOS MLS system with `PRIV_SU` enabled, you must be `root` to use this command.

FILES

`/usr/lib/tp/tpdaemon`	Tape daemon executable file
`/usr/spool/tape/trace`	Trace files for tape devices

SEE ALSO

`oper`(8), `tpclr`(8), `tpconfig`(8), `tpfrls`(8), `tpset`(8), `tpgstat`(8), `tpmls`(8), `tpu`(8)

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

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

General UNICOS System Administration, Cray Research publication SG–2301

Tape Subsystem Administration, Cray Research publication SG–2307

NAME

`tpdaemon` – Starts tape daemon

SYNOPSIS

`/usr/lib/tp/tpdaemon` [`-b`] [`-c`] [`-C` *config_file*] [`-d`] [`-l`] [`-m` *devs*] [`-r`]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `tpdaemon` command starts the tape daemon. It provides the routing and control of the various components used in tape resource management, device management, volume mounts and dismounts through operator communication or autoloader requests, label processing, volume switching, accounting, security, and error recovery. By default, tape configuration is defined as part of the tape daemon start-up.

The `tpdaemon` command accepts the following options:

`-b` Instructs the tape daemon to bypass all configuration tasks. `tpdaemon` uses the existing tape configuration instead of configuring the tape subsystem based on the tape configuration file.

If you specify this option and have not previously defined the configuration by using the `tpdaemon` command without the `-b` option, `tpdaemon` terminates with an error. If you omit the `-b` option, `tpdaemon` reads and processes a tape configuration file, provides the configuration information to the kernel and tape I/O processors (IOPs), creates the tape device nodes, and configures the channels and control units as specified in the tape configuration file. Options `-b` and `-C` are mutually exclusive.

`-c` Collects tape user data for accounting.

Options `-b` and `-C` are mutually exclusive.

`-C` *config_file* Specifies the file that contains the tape configuration. If you do not specify the `-C` option, the tape daemon uses the `/etc/config/text_tapeconfig` file. If the `tpinit`(8) command does not find this file, it terminates and returns an error message.

`-d` Does not initiate or verify the devices, channels, control units, or banks in the tape configuration file.

`-l` Locks the tape daemon into memory so that it will not be swapped out.

`-m` *devs* Preallocates heap memory for *devs* number of devices. If the tape daemon is locked in memory and needs to grow, using the `-m` option reduces the probability of the tape daemon being swapped out of memory.

`-r`	Removes all trace files before creating new ones. If you do not specify this option, the existing trace files are used. If possible, the old trace files are linked to files in the directory specified in `/etc/config/text_tapeconfig` by the `tape_daemon_trace_savefile_prefix`.

For more information on starting the tape daemon on a UNICOS system, see the section on enhanced security in *General UNICOS System Administration*, Cray Research publication SG–2301.

NOTES

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

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

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

FILES

`/dev/tpddem`	Tape daemon interface device
`/dev/tape/`*device_name*	Tape device file
`/etc/config/text_tapeconfig`	Tape subsystem configuration file
`/usr/spool/tape/trace.`*bmxXXX*	Trace file for tape device. The block multiplexer (BMX) number is associated with the drive in the `tpstat`(1) display.

SEE ALSO

`tpconf`(8), `tpdstop`(8), `tpinit`(8)

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

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

General UNICOS System Administration, Cray Research publication SG–2301

Tape Subsystem Administration, Cray Research publication SG–2307

NAME

`tpdev` – Displays current tape equipment configuration

SYNOPSIS

`/etc/tpdev`

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `tpdev` command displays the status of tape devices and associated components of the tape data path. The following information is displayed in the status output (control unit numbers are hexadecimal; all other numbers are octal):

`dev_name`	Specifies device name
`dev_grp`	Specifies device group name
`bk`	Specifies bank number
`ord`	Specifies device order
`did`	Specifies device ID number
`dvst`	Specifies the device status
`cu`	Specifies the control unit ID number
`cust`	Specifies the control unit status
`ipcc`	(IOS-E only) Specifies a 1-digit (`i`) IOS cluster number, a 1-digit (`p`) IOP number, and a 2-digit (`cc`) channel number
`rrnncc`	(GigaRing based systems only) Specifies a 2-digit (`r`) ring number, a 2-digit (`n`) node number, and a 2-digit (`cc`) channel number
`chst`	Specifies the channel status
`loader`	Specifies the media loader, if any, for the device

NOTES

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

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

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

EXAMPLES

The following display (IOS-E only) is an example of `tpdev` output:

```
dev_name  dev_grp  bk ord  did  dvst  cu   cust  ipcc   chst  loader
130       CART      0   0    0  down   1   up    0321   up    Operator
131       CART      0   1    1  down   1   up    0321   up    Operator
210       TAPE      1   4    0  down   2   up    0321   up    Operator
211       TAPE      1   5    1  down   2   up    0321   up    Operator
170       3490      2  10    0  down   A   up    0326   up    Operator
171       3490      2  11    1  down   A   up    0326   up    Operator
300       SILO      3  14    0  up     6   up    0326   up    stksun
301       SILO      3  15    1  down   6   up    0326   up    stksun
B00       WOLF      8  30    0  up     A   up    0327   up    wolfy
B01       WOLF      8  31    1  up     A   up    0327   up    wolfy
```

The following display (GigaRing based systems only) is an example of `tpdev` output:

```
dev_name  dev_grp  bk ord  did  dvst  cu   cust  rrnncc   chst  loader
s4890s0   STK4890   1   0    0  down  01   up    000101   up    panther
s4890s1   STK4890   1   1    8  down  01   up    000101   up    panther
d4000s0   DLT4000   1   2    0  down  01   up    000101   up    panther
d4000s1   DLT4000   1   3    8  down  01   up    000101   up    panther
3490s0    IBM3490E  6   4    0  up    06   up    000106   up    ibm
3490s1    IBM3490E  6   5    1  up    06   up    000106   up    ibm
d5649JX   DAT      17   6    0  down  07   up    010107   up    Operator
s9490s0   STK9490  16   7    0  down  06   up    010106   up    wolfy
s9490s1   STK9490  16  10    8  down  06   up    010106   up    wolfy
s9490s2   STK9490  16  11    0  down  06   up    010106   up    wolfy
s9490s3   STK9490  16  12    8  down  06   up    010106   up    wolfy
3590s0    IBM3590  10  13    0  down  00   up    010100   up    ibm
3590s1    IBM3590  12  14    8  down  02   up    010102   up    ibm
ssd3_s0   STKSD3   11  15    8  down  01   up    010101   up    wolfy
ssd3_s1   STKSD3   13  16    8  down  03   up    010103   up    wolfy
```

SEE ALSO

`tpconfig`(8)

General UNICOS System Administration, Cray Research publication SG– 2301

Tape Subsystem Administration, Cray Research publication SG– 2307

NAME

`tpdstop` – Stops tape daemon

SYNOPSIS

```
/etc/tpdstop
```

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `tpdstop` command stops the tape daemon, which terminates in an orderly fashion if the operator uses `tpdstop` to stop it. If the tape daemon does not terminate after the `tpdstop` command, the system administrator may send the tape daemon an interrupt signal by using the `kill`(1) command (*pid* is the process ID of the tape daemon), as follows:

```
kill -2 pid
```

When the tape daemon catches the interrupt signal, it terminates in a less orderly way than if you used `tpdstop`.

If the tape daemon still refuses to terminate after being sent the interrupt signal, the super user may kill it in the following way (again, *pid* is the process ID of the tape daemon):

```
kill -9 pid
```

NOTES

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

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

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

SEE ALSO

`tpdaemon`(8)

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

General UNICOS System Administration, Cray Research publication SG–2301

Tape Subsystem Administration, Cray Research publication SG–2307

NAME

`tpformat` – Formats an ER90 tape volume

SYNOPSIS

`/etc/tpformat` [`-C`] [`-D` *device-name*] [`-e`] [`-f` *format-ID*] [`-g` *device-group-name*] [`-l` *length*] [`-n` *number-of-A-partitions* [`:`*number-of-B-partitions*]] [`-q`] [`-s` *A-partition-size*[`:`*B-partition-size*]] [`-u`] `-v` *volume-ID* [`-w`] [`-z`]

IMPLEMENTATION

All Cray Research systems except CRAY J90se systems

DESCRIPTION

The `tpformat` command reserves the specified resource, mounts the requested volume, and issues the volume format request to the ER90 device. After the format is completed, it releases the reserved resource. Use this command to preformat a volume or to specify that the volume be formatted during normal write operations. Preformatting formats the entire volume from beginning-of-tape (BOT) to end-of-tape (EOT), creates the specified partitions, and creates the system zones (if requested).

The `tpformat` command requires the following argument:

`-v` *volume-ID*
Specifies the volume identifier that appears on the physical, outer label of the tape to be formatted. This identifier is used to communicate with the system operator.

The `tpformat` command accepts the following options:

`-C` Specifies that the character-special tape interface be used. The character-special tape interface allows access to a tape device without using the tape daemon. You are responsible for allocating a resource for the tape format and mounting a volume on a drive. You must also specify the device to which the format request is issued by using the `-D` option. By default, the tape daemon-assisted interface is used.

`-D` *device-name*
Specifies the name of the device to use for the tape format.

If the tape daemon-assisted interface is used, the tape daemon will issue a mount request to the specified volume. If the requested device is busy (assigned to another user), the request will be queued until the device is available. If the volume serial number (VSN) requested is premounted and idle on another drive, that drive will be used instead of the drive requested. When automatic volume recognition (AVR) is enabled, this option is not available.

`-e` Specifies that the ER90 attempt to minimize the amount of system zone discontinuities in a partition. If the ER90 device determines that a partition should be created after a system zone, the previous partition is extended to the system zone dividing the two partitions. Options `-e` and `-w` are mutually exclusive. If neither option is specified, partitions are allowed to span system zones.

`-f` *format-ID*
: Specifies the identifier to be recorded on the tape during the volume format. The format identifier must not be longer than 6 alphanumeric characters. If you do not specify a format ID, the volume identifier specified with the `-v` option is recorded on the tape.

`-g` *device-group-name*
: Specifies that the volume be mounted on the device belonging to the group *device-group-name.* If you omit *device-group-name*, it defaults to an installation-specified name. The device group name (*dgn*) field of the `tpstat`(1) command shows the allowable parameters.

`-l` *length*
: Specifies the length, in double frames, between system zones. The length specified must be in the range 0x842 through 0xFFFFFF. You should not change this length from the device default.

`-n` *number-of-A-partitions* [`:`*number-of-B-partitions*]
: Specifies the number of A partitions and the number of B partitions that should be formatted. The number of A partitions specified must be in the range 1 through 255; the size is specified by using the `-s` option. These partitions are formatted on the tape until all partitions have been created or until the end of the tape is detected. If no B partitions are specified (or if *number-of-B-partitions* is specified as 0), the ER90 device formats the tape with A partitions until the EOT is detected. If the EOT is detected after formatting all A partitions and no B partitions are specified, the system continues formatting with A partitions. The default is to create one A partition.

  The number of B partitions specified must be in the range 0 through 255. B partitions are created following the last A partition. If one B partition is requested with a size of 0 (as specified with the `-s` option), the volume is formatted with one B partition spanning the remainder of the volume. If you specify more than one B partition, the volume is formatted with B partitions until all partitions are formatted or until the EOT is detected. The default is 0 B partitions.

  If the end of the volume is not detected after creating the B partitions, formatting continues, beginning again with A partitions.

`-q`
: Specifies that the volume should be formatted during normal write operations. The default is to format the entire volume. When using this option, you cannot specify multiple formats.

`-s` *A-partition-size*[`:`*B-partition-size*]
: Specifies the size (in millions of bytes) of the partitions. You must specify the number of partitions by using the `-n` option for this option to be valid. The size of the partition must be in the range 0, 0xF0 through 0x1312D00 (240 through 20,000,000 bytes).

  If you specify a size of 0 for the A partition, one partition is created spanning the length of the volume. Any size specified for the B partition is then not valid.

  If you specify 0 for the B partition size, one B partition is created spanning the length of the tape remaining after the A partitions.

`-u`
: Disables tape unload at release time. A tape usually is unloaded automatically after the format completes.

`-w`	Specifies that the ER90 attempt to minimize the number of system zone discontinuities within a partition. If the ER90 device determines that a partition should be created after a system zone, the previous partition is not extended to the system zone dividing the two partitions. Instead, the area between the previous partition and the system zone is wasted. Options `-e` and `-w` are mutually exclusive. If you omit both options, the partitions may span system zones.
`-z`	Specifies that the volume should be formatted without system zones.

NOTES

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

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

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

If the format request does not complete successfully, the integrity of the cassette cannot be guaranteed. You should not use the cassette unless it is reformatted successfully.

If the tape error message TM297 is received, the ER90 device was not able to complete the format request within the time-out period specified in the tape configuration file. This time-out period is specified with the `long_timeout` parameter for ER90 over IPI devices; it is specified with the `format_timeout` parameter for ER90 over HIPPI devices. The tape is not usable if this message is received. To correct the error condition, you can increase the time-out period in the tape configuration file, restart the tape daemon, and reissue the tape format request.

If a volume is formatted without system zones, the volume is positioned to the beginning-of-tape (BOT) or the end-of-tape (EOT) when it is unloaded. It could take up to 185 seconds to complete this unload. By using the default system zone spacing, you can reduce this time to approximately 16 seconds for a small cassette, 21 seconds for a medium cassette, and 24 seconds for a large cassette.

The actual tape format may not be equivalent to the requested format as described in the following paragraphs.

For each partition, the ER90 device creates a beginning-of-media (BOM) marker and reserves the end-of-media warning (EMW) area following the end of the partition. These areas use approximately 42 million bytes of the tape per partition in addition to the partition size requested. Of this 42 million bytes, 12 million bytes are used for the EMW area and 30 million bytes are used for the BOM marker. 8.4 million bytes of the EMW area are available to the user. (The tape daemon allows the user to write into this area by default.)

System zones use additional tape space. For example, for a small cassette in which 3 system zones are created, 809.7 million bytes are used for system zones.

EXAMPLES

Example 1: This example formats a tape with one partition spanning the length of the tape. The format ID recorded on the cassette defaults to the volume ID specified by using the `-v` option.

```
tpformat -v 000011 -g ER90
```

Example 2: This example divides a 25-Gbyte cassette into three partitions of equal length. Each partition consists of approximately 0x208d Mbytes. The format ID recorded on the tape is ER9011. The tape is left on the drive after the format completes because the `-u` option is specified.

```
tpformat -v 000011 -f ER9011 -n 3 -s 0x208d -u -g ER90
```

Example 3: This example formats a cassette into one partition consisting of 0xF0 Mbytes and one partition spanning the remainder of the tape. No system zones are created.

```
tpformat -v 000011 -n 1:1 -s 0xF0:0 -g ER90 -z
```

Example 4: This example requests that one partition consisting of 240 Mbytes be formatted on a 25-Gbyte tape. Because unformatted tape remains after formatting one 240 Mbyte partition, the ER90 device continues creating partitions that consist of 240 Mbytes until the end of tape is detected. There will be approximately 90 partitions on the tape.

```
tpformat -v 000011 -n 1 -s 240 -g ER90
```

Example 5: This example shows how to format a volume when you are using the character-special tape interface. The `tpformat` command formats the volume mounted on device `er92` with 2 partitions consisting of 3 Gbytes and 9 partitions consisting of approximately 2 to 6 bytes.

```
tpformat -f ER4810 -s 3000:2000 -n 2:9 -C -D er92
```

SEE ALSO

General UNICOS System Administration, Cray Research publication SG–2301

Tape Subsystem Administration, Cray Research publication SG–2307

NAME

`tpfrls` – Forcibly releases tape reservation and associated devices

SYNOPSIS

`/etc/tpfrls` *user jobid*

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `tpfrls` command lets the operator release the tape reservations made by a user. To obtain the status of all users' reservations, use the `tpgstat`(8) command.

You must specify the following operands with `tpfrls`:

user	Indicates the user ID of the user whose resources will be released.
jobid	Indicates the job ID of the user whose reservation will be released.

`tpfrls` also clears all active tape devices and kills the user process using the tape devices.

NOTES

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

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

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

EXIT STATUS

If `tpfrls` 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.

SEE ALSO

`tpgstat`(8)

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

General UNICOS System Administration, Cray Research publication SG–2301

Tape Subsystem Administration, Cray Research publication SG–2307

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

NAME

`tpgstat` – Displays user reservation status for all users

SYNOPSIS

`/etc/tpgstat [-a]`

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `tpgstat` command displays the reservation status for each device group in the system for each tape user. The `tpgstat` command accepts the following option:

`-a`	Restricts output to each device group that has been reserved or that is being used by a tape user in the system.

The `tpgstat` command provides the following status:

`user`	User ID.
`jobid`	Job ID.
`dgn`	Device group name.
`w`	If the user is waiting for a device, an asterisk (`*`) is displayed.
`rsvd`	Number of devices reserved.
`used`	Number of devices used.
`mins`	Number of minutes the user has the resources reserved.
`NQSid`	Specifies Network Queuing System (NQS) ID if the reserved resources have been submitted through NQS. You may use this ID for `qstat`(1) or `qdel`(1).

NOTES

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

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

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

Any devices not reserved by other users show up as being reserved by `tpdaemon`.

EXIT STATUS

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

EXAMPLES

In the following example display, user `lfa` is waiting for a device that is type `TAPE`:

```
user        job id   dgn    w   rsvd   used   mins   NQSid
tpdaemon      3      TAPE         2      2     110
              3      CART         2      1     110
lfa          362     TAPE   *     1      1      37
             362     CART         0      0      37
backup       302     TAPE         0      0      19
             302     CART         1      1      19
dkl          379     TAPE         1      1      15
             379     CART         0      0      15
jwa          251     TAPE         1      0      0
             251     CART         0      0      0
cym          667     TAPE         0      0      1     4082.sn1101
             667     CART         1      1      1     4082.sn1101
```

SEE ALSO

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

General UNICOS System Administration, Cray Research publication SG–2301

Tape Subsystem Administration, Cray Research publication SG–2307

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

NAME

`tpinit` – Initializes the tape subsystem

SYNOPSIS

`/etc/tpinit` [`-C` *config_file*] [`-d`]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `tpinit` command initializes the tape subsystem. It reads and processes the tape configuration file, provides configuration information to the kernel and tape I/O processors (IOPs), creates the tape device nodes, and, unless instructed not to do so, configures the channels and control units or slaves as specified in the tape configuration file. The command is executed automatically at system startup if tapes are defined.

You cannot run `tpinit` if a tape device node is open or if the tape daemon is active.

The `tpinit` command accepts the following options:

`-C` *config_file* Specifies the file that contains the tape configuration. If you omit the `-C` option, the command uses the `/etc/config/text_tapeconfig` file as the tape configuration file.

`-d` Instructs the command to leave the tape hardware in its existing state rather than attempt to change the state of the hardware. If you omit the `-d` option, `tpinit` configures the channels, banks, and control units or slaves as specified in the tape configuration file.

NOTES

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

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

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

FILES

`/dev/tpddem`	Tape daemon interface
`/dev/tape/`*device_name*	Tape device node
`/etc/config/text_tapeconfig`	Tape subsystem configuration file

SEE ALSO

`tpconfig`(8), `tpdaemon`(8), `tpdstop`(8)

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

Tape Subsystem Administration, Cray Research publication SG–2307

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

NAME

`tplabel` – Labels a magnetic tape

SYNOPSIS

`/etc/tplabel` [`-C`] [`-D` *device_name*] [`-g` *device_group_name*] [`-l` *label_type*] [`-I`] [`-o` *owner_id*] [`-s` *system_code*] [`-u`] [`-v` *int_vid*[=*ext_vid*][=*format_id*][/*partition*]]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `tplabel` command labels tapes and may perform other functions depending on the interface being used.

When you are using the character-special tape interface and want to label a tape, you must mount the desired tape on the target device prior to using a `tplabel` command. After the label is written, the tape is unloaded.

When you are using the tape daemon-assisted interface and issue a `tplabel` command, the requested resource must be configured up. The `tplabel` command reserves the specified resource, requests the tape mount, labels the tape, and finally releases the resource.

The `tplabel` command accepts the following options:

`-C` Specifies that the character-special tape interface be used. The character-special tape interface allows access to a tape device without using the tape daemon. You are responsible for allocating a resource for the volume labeling and mounting a volume on a drive. You must also specify the device to which the label request is issued by using the `-D` option. By default, the tape daemon-assisted interface is used.

`-D` *device_name*
Specifies the name of the device to use for labeling the tape volume.

If the tape daemon-assisted interface is used, the tape daemon will issue a mount request to the specified volume. If the requested device is busy (assigned to another user), the request will be queued until the device is available. If the volume serial number (VSN) requested is premounted and idle on another drive, that drive will be used instead of the drive requested. When automatic volume recognition (AVR) is enabled, this option is not available.

`-g` *device_group_name*
Specifies that the volume be mounted on a device belonging to the group *device_group_name*. If you do not specify the device group name, the volume will be mounted on a device that belongs to the installation-defined default. `tpstat`(1) displays the valid device group names.

`-l` *label_type* Specifies the label type to create. The label type must be one of the following:

`al`	ANSI label
`nl`	Not labeled
`sl`	IBM standard label

If you omit this option, a nonlabeled tape will be created. For nonlabeled tapes, three tape marks are written at the beginning of the tape.

`-I` ER90 volumes only. Specifies that mount verification that uses the *format_id* should be bypassed. To use this option, the user must have root user bypass label or tape manager permission.

`-o` *owner_id* Specifies the owner ID field in the VOL1 header label. The *owner_id* can consist of up to 14 characters. This value defaults to `CRI/UNICOS`.

`-s` *system_code* Specifies the system code field in the HDR1 and EOF1 label. The *system_code* can consist of up to 13 characters. This value defaults to `CRI/UNICOS`.

`-u` Disables the tape unload at release time. If you omit this option, the volume will unload after the format completes.

`-v` *int_vid*[=*ext_vid*][=*format_id*][/*partition*]
Specifies the volume ID of the tape. You can specify the *volume_id* in lowercase or uppercase letters; however, the system converts the internal and external volume IDs to uppercase letters.

int_vid Specifies the ID that will be used in the VOL1 label for `al` or `sl` label type. This ID is not used for volume verification; therefore, it may differ from the *external_volume_id*.

ext_vid Specifies the identifier that appears on the physical, outer label of the tape reel. This ID is used to communicate with the system operator. If you omit an external volume identifier, the default is the internal volume identifier.

format_id ER90 volumes only. Specifies the volume identifier that the ER90 device records on the tape during volume initialization. This ID is used for mount verification. If you omit a format identifier, the external volume identifier is used. If an external volume identifier is not specified, the internal volume identifier is used. If you omit an external volume identifier and want a format identifier, the format identifier follows two equal signs (*internal_volume_id*==*format_id*).

/partition ER90 volumes only. Specifies the partition number to which the volume should be positioned. The partition number must be in the range of 0 through 1023. If you omit the partition number, the default is 0.

NOTES

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

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

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

EXAMPLES

Example 1: This example creates an IBM standard label on a volume with the external identifier of EXT011. The VOL1 label contains the volume ID `000011` and the owner ID `CRI`. The system code is `UNICOS`.

```
tplabel -v 000011=EXT011 -l sl -o CRI -s UNICOS -g CART
```

Example 2: This example creates a nonlabeled tape on the third partition of an ER90 volume. The volume must have a format ID of `FMT011`. Three tape marks are created at the beginning of the volume. The `-u` option specifies that the volume is not unloaded.

```
tplabel -v 000011==FMT011/2 -l nl -u -g ER90
```

Example 3: This example shows how to create an ANSI labeled tape on the volume mounted on device `300` when you are using the character-special tape interface. The `tplabel` command expects the volume to be mounted on device `300`.

```
tplabel -C -D 300 -l al -o CRI -s UNICOS -v  001940
```

Example 4: This example shows how to create IBM standard labels on partitions 2 through 5 on the volume mounted on `er93` when you are using the character-special tape interface. The `tplabel` command expects the volume to be mounted on device `er93`. The default owner system codes, `CRI/UNICOS`, are used.

```
tplabel -C -D er93 -p 2-5 -v 004810 -l sl
```

SEE ALSO

`tpdaemon`(8)

General UNICOS System Administration, Cray Research publication SG–2301

Tape Subsystem Administration, Cray Research publication SG–2307

NAME

`tpmls` – Displays loader status

SYNOPSIS

```
/etc/tpmls
```

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `tpmls` command displays the status of the tape loaders in the system. The tape loader configuration information is built from the `/etc/config/text_tapeconfig` file, and the tape daemon updates it during normal processing. You can use the `tpconfig`(8) command to configure loaders up or down or to place them in attended or unattended mode of operation.

The `tpmls` command display contains the following information:

`loader`	Name by which devices and loader requests are associated.
`type`	The type of loader. Currently supported loader types are `OPERATOR`, `STKACS`, and `IBMSCR`.
`status`	`UP`, `DOWN`, `MANUAL`, `CNFPND`, `SYSDWN`, `SYSMAN`, or `UNKNOWN`. Any loader configured down has its drives configured down.
`m`	Mode of operation. Possible values are as follows:

`A`	Attended
`U`	Unattended (auto mode)
`S`	Full system mode (IBM scratch loader)
`M`	Manual mode

`server`	Name or collection of front-end IDs to which messages may be sent for service.
`old`	Number of online drives.
`m_pnd`	Number of mounts pending for the loader.
`d_pnd`	Number of dismounts pending for the loader.
`r_qd`	Requests queued waiting for a drive serviced by this loader.
`comp`	Number of mounts completed.
`avg`	Average time to complete mount request.

NOTES

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

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

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

EXAMPLES

The following example shows a typical display created by executing `tpmls`:

```
  loader     type     status m  server   old  m_pnd  d_pnd  r_qd  comp  avg
 ========  ========  ====== = ========  ===  =====  =====  ====  ====  ===
 Operator  OPERATOR  UP     A UNICOS      0      0      0     0     0    0(sec)
 stkvm     STKACS    DOWN   A V3          0      0      0     0     0    0(sec)
 stkmvs    STKACS    DOWN   A M4          0      0      0     0     0    0(sec)
 stksun    STKACS    UP     A robot       1      0      0     0     8   41(sec)
 wolfy     STKACS    UP     A stkwolf     4      0      1     0   166   31(sec)
```

FILES

`/etc/config/text_tapeconfig`	Tape subsystem configuration file

SEE ALSO

`tpconfig`(8), `tpdev`(8)

`text_tapeconfig`(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

Tape Subsystem Administration, Cray Research publication SG–2307

NAME

`tpmql` – Displays tape daemon mount request queue list

SYNOPSIS

`/etc/tpmql`

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `tpmql` program displays the current tape mount request list for all users who have completed initial mount processing and have a mount request pending.

If you use the `-O` option on the `tpmnt`(1) command to specify an offset to a volume identifier, the output of the `tpmql` command is no longer accurate. The `tpmql` command displays the entire list of volume identifiers. If an offset has been set, the list of identifiers from `tpmql` may no longer correspond to the order in which the tapes have been accessed using the `tpmnt`(1) command.

NOTES

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

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

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

EXAMPLES

The following example shows the output of `tpmql`:

```
userid     jobid     reqid     ring     Vsn list
jas        19        14        in       UP0007
dja        208       15        out      ISCAL
bar        549       27        i/o      BARRRY1
```

In the `ring` field, `i/o` indicates that the user has omitted the `-r` argument on the `tpmnt`(1) command and the system default is to accept the tape with the ring in or out.

The fields in the display have the following meanings:

`userid`	The user ID of the job running tapes
`jobid`	The job ID of the job running tapes
`reqid`	The request ID (internal to the tape daemon)

`ring`	The ring status associated with the `Vsn list`
`Vsn list`	The volume serial number (VSN) of the next volume to be mounted

SEE ALSO

`tpapm`(8)

General UNICOS System Administration, Cray Research publication SG–2301

Tape Subsystem Administration, Cray Research publication SG–2307

NAME

`tpscr` – Returns volumes to loader scratch pool

SYNOPSIS

`/etc/tpscr -v` *vsnlist loader_name*
`/etc/tpscr -V` *file loader_name*

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `tpscr` command returns volumes allocated by a user to the loader scratch pool. You must specify the following options and operands with `tpscr` (you cannot use the `-v` and `-V` options together):

`-v` *vsnlist*	Specifies a list of volume serial numbers (VSNs).
`-V` *file*	Specifies a file that contains the VSNs.
loader_name	The name of the loader to which the allocated volumes belong; this name must match one of the loader names in the `/etc/config/text_tapeconfig` file.

The VSN list and the VSN file follow the same rules as those used with the `tpmnt`(1) command.

NOTES

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

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

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

The `define DEF_SCR_ACTION` in the `/etc/config/text_tapeconfig` file controls whether VSNs allocated by the loader are returned to the scratch pool when a user issues a `rls`(1) command.

MESSAGES

A scratch request is issued for each VSN specified. Those scratch requests that cannot be completed are displayed with the VSN and the return code from the scratch request.

FILES

`/etc/config/text_tapeconfig`	Tape subsystem configuration file

SEE ALSO

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

`text_tapeconfig`(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

Tape Subsystem Administration, Cray Research publication SG–2307

NAME

`tpset` – Sets features for the tape daemon

SYNOPSIS

`/etc/tpset` [`-a` *state*] [`-c` *state*] [`-f` *state*] [`-g` *device_group*] [`-M` *number*] [`-o` *operator*]
[`-O` *state*] [`-T` *state*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `tpset` command sets features for the tape daemon. It changes the status of automatic volume recognition (AVR), the status of front-end servicing (FES), the status of the Cray/REELlibrarian (CRL) system, the status of overcommitted mount requests, the status of tracing for the tape daemon, or the destination of tape operator messages issued by the tape daemon. All of these features except tracing can be set in the tape configuration file. Tracing is always available for use in problem resolution unless you turn it off by specifying the `-t` option.

The `tpset` command can be used by both operators and system administrators. For information about the tape daemon, see the `tpdaemon`(8) man page.

If you omit the options, `tpset` returns the current status of all features. You may use the `-a`, `-c`, and `-f` options on only a quiet tape subsystem;

The `tpset` command accepts the following options:

`-a` *state* Specifies whether AVR is available. Enter one of the following for *state*:

`on`	Enables AVR.
`off`	Disables AVR. Overcommitted mount requests must be disabled before you can disable AVR.

`-c` *state* Specifies whether CRL processing is available. Enter one of the following for *state*:

`mandatory`	Requires CRL processing for all `tpmnt`(1) requests.
`on`	Enables CRL processing only when the `-X` option to `tpmnt`(1) is used.
`off`	Disables CRL processing.

`-f` *state* Specifies whether FES is available. Enter one of the following for *state*:

`on`	Enables FES.
`off`	Disables FES.

`-M` *number* Specifies the maximum number of overcommitted mount requests that the tape subsystem can issue.

When the actual number of tape mount requests exceeds this number, the system stops processing requests until one or more of the already overcommitted mount requests are satisfied.

`-o` *operator*
: Changes the destination that receives operator messages from the tape daemon.

Enter one of the following for *operator*:

`UNICOS`	Routes operator messages to the message daemon.
station	Specifies a 32-character station ID, which signifies a servicing front end for your UNICOS system.

`-O` *state*
: Specifies whether the number of current mount requests can exceed the number of available tape drives.

Enter one of the following for *state*:

`on`	Enables the use of overcommitted mount requests only when the AVR is enabled.
`off`	Disables the use of overcommitted mount requests. You must wait to specify `off` until all overcommitted mount requests are satisfied.

An operator must respond to the regular mount requests before mounting tapes for overcommitted mount requests. When an unassigned tape drive becomes available, an operator mounts an overcommitted tape, and the system then assigns the drive to the user's job. Overcommitted mount requests work for jobs that need only one tape mount or that are requesting the last of a series of tape mounts.

If you omit the `-g` option, overcommitted mount requests are available to all device groups that have AVR enabled and that use only operator-mounted tapes. If you specify the `-g` option, overcommitted mount requests are available only to the specified device group.

`-T` *state*
: Specifies whether tracing is enabled for the tape daemon. Enter one of the following for *state*:

`on`	Enables tracing. The default is `on`.
`off`	Disables tracing.

The new state does not affect child processes executing at the time of the change.

NOTES

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

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

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

SEE ALSO

`tpdaemon`(8)

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

General UNICOS System Administration, Cray Research publication SG–2301

Tape Subsystem Administration, Cray Research publication SG–2307

NAME

`tpu` – Unloads tape drives

SYNOPSIS

`/etc/tpu` [`-g`] *dvn*[:*dvn*. . .]
`/etc/tpu` [`-g`] `all`

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The system operator uses the `tpu` command to unload tapes. This command has no effect on a tape that is currently in use. It is most useful for unloading tapes and freeing tape drives on systems running with automatic volume recognition (AVR).

The `tpu` command accepts the following option and operands:

`-g`	Specifies that the current *no-unload* (cannot be unloaded) status will be disregarded. `tpu` does not affect a tape drive with no-unload status unless you specify the `-g` option. See the `tpconfig`(8) command for more information about tapes with the no-unload status.
dvn	Specifies the device names of the tape drives to be unloaded.
`all`	Specifies that all tape drives configured up and not in use will be unloaded.

NOTES

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

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

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

SEE ALSO

`tpconfig`(8)

General UNICOS System Administration, Cray Research publication SG–2301

Tape Subsystem Administration, Cray Research publication SG–2307

NAME

`traceroute` – Prints the route that packets take to network host

SYNOPSIS

`/etc/traceroute` [`-d`] [`-f`] [`-m` *max_ttl*] [`-n`] [`-p` *port*] [`-q` *nqueries*] [`-r`] [`-s` *src_addr*] [`-S` *tos*] [`-v`] [`-w` *waittime*] *host* [*packetsize*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `traceroute` command traces the route that the Internet Protocol (IP) uses when sending a packet from the local system to the given remote host. This command depends on aspects of IP that do not exist in every environment; therefore, it does not display hops through a system whose IP does not support this feature. `traceroute` continues until the remote host is reached. Hosts whose IP does not support the necessary feature show up as hops displayed as all * (asterisks). `traceroute` traces the route only from the local system to the remote host.

The Internet is a large and complex aggregation of network hardware connected by gateways. The `traceroute` command uses the IP protocol time-to-live (ttl) field and tries to elicit an `ICMP TIME_EXCEEDED` response from each gateway along the path to some host.

The only mandatory argument in the `traceroute` command is the destination host name or IP number. The name may be a source routed address; the format is *@hop1@hop2*`. . .`*@dst*. By default, a loose source route is used; if the route is preceded by the `!` character, a strict source route is generated. The default probe datagram consists of 38 bytes; to increase this size, specify a packet size (in bytes) after the destination host name.

The `traceroute` command accepts the following options:

`-d`	Turns on the `SO_DEBUG` option on the socket.
`-f`	Turns on the `DONT_FRAGMENT` bit in the IP header.
`-m` *max_ttl*	Sets the maximum time-to-live (maximum number of hops) used in outgoing probe packets. The default is 30 hops, which is also the default used for TCP connections.
`-n`	Prints hop addresses numerically, rather than symbolically and numerically. This saves a name server address-to-name lookup for each gateway found on the path.
`-p` *port*	Sets the base user datagram protocol (UDP) port number used in probes. Default is 33434. `traceroute` hopes that nothing is listening on UDP ports *base* to *base*+*nhops*–1 at the destination host (so that an `ICMP PORT_UNREACHABLE` message is returned to terminate the route tracing). If a daemon is listening on a port in the default range, use this option to select an unused port range.
`-q` *nqueries*	Specifies the number of probes allowed.

`-r`	Bypasses the normal routing tables and sends directly to a host on an attached network. If the host is not on a directly attached network, an error is returned. Use this option to ping a local host through an interface that has no route through it (for example, after the interface was dropped by `gated`(8)).
`-s` *src_addr*	Uses the IP address *src_addr* (which must be given as an IP number, not a host name) as the source address in outgoing probe packets. On hosts that have more than one IP address, you can use this option to force the source address to be something other than the IP address of the interface on which the probe packet is sent. If the IP address is not one of this machine's interface addresses, an error is returned and nothing is sent.
`-S` *tos*	Sets the Type-of-Service (*tos*) in probe packets to the following value. Default is 0. The value may be an integer in the range 0 to 255, or a symbolic TOS name found in the `/etc/iptos` file. You can use this option to determine whether different types of service result in different paths. Not all values of *tos* are legal or meaningful; for complete information, see the IP specification for definitions. Values may be either numeric or names listed in the `/etc/iptos` file; useful values include `-S delay`, `-S 16` (low delay), and `-S throughput` or `-S 8` (high throughput). For historic compatibility, the `-t` option is accepted as a synonym.
`-v`	Specifies verbose output. When you specify `-v`, all received ICMP packets other than `TIME_EXCEEDED` and `UNREACHABLE` are listed.
`-w` *waittime*	Sets the time, in seconds, to wait for a response to a probe. The default is 3 seconds.
host	Specifies destination host.
packetsize	Specifies a packet size that is an alternative to the default size of 38 bytes.

The `traceroute` command tries to trace the route that an IP packet would follow to some Internet host by launching UDP probe packets with a small ttl (time to live) and then by listening for an ICMP `time exceeded` reply from a gateway. `traceroute` starts the probes with a ttl of 1, and it increases by 1 until it receives an ICMP `port unreachable` message (which means `traceroute` reached *host*) or hits a maximum (which defaults to 30 hops and can be changed by using the `-m` option). Three probes (change the number by using the `-q` option) are sent at each ttl setting, and a line is printed showing the ttl, the gateway address, and round-trip time of each probe. If the probe answers come from different gateways, the address of each responding system is printed. If no response occurs within a 3-second time-out interval (changed with the `-w` option), an asterisk character (*) is printed for that probe.

The destination host should not process the UDP probe packets; therefore, the destination port is set to an unlikely value. If someone on the destination is using that value, it can be changed with the `-p` option.

CAUTIONS

This program is for use in network testing, measurement, and management. It should be used primarily for manual fault isolation. Because of the load it may impose on the network, it is unwise to use `traceroute` during normal operations or from automated scripts.

EXAMPLES

Example 1: A sample use of `traceroute`, including output, is as follows.

```
[yak 71]% traceroute nis.nsf.net
traceroute to nis.nsf.net (35.1.1.48), 30 hops max, 56 byte packet
 1  helios.ee.lbl.gov (128.3.112.1)  19 ms  19 ms  0 ms (19.0/12.667)
 2  lilac-dmc.Berkeley.EDU (128.32.216.1)  39 ms  39 ms  19 ms (39.0/32.333)
 3  lilac-dmc.Berkeley.EDU (128.32.216.1)  39 ms  39 ms  19 ms (39.0/32.333)
 4  ccngw-ner-cc.Berkeley.EDU (128.32.136.23)  39 ms  40 ms  39 ms (39.0/39.333)
 5  ccn-nerif22.Berkeley.EDU (128.32.168.22)  39 ms  39 ms  39 ms (39.0/39.0)
 6  128.32.197.4 (128.32.197.4)  40 ms  59 ms  59 ms (59.0/52.667)
 7  131.119.2.5 (131.119.2.5)  59 ms  59 ms  59 ms (59.0/59.0)
 8  129.140.70.13 (129.140.70.13)  99 ms  99 ms  80 ms (99/92.667)
 9  129.140.71.6 (129.140.71.6)  139 ms  239 ms  319 ms (239.0/232.333)
10  129.140.81.7 (129.140.81.7)  220 ms  199 ms  199 ms (199.0/206.000)
11  nic.merit.edu (35.1.1.48)  239 ms  239 ms  239 ms (239.0/239.0)
```

Lines 2 and 3 are the same. This is due to a bug in the kernel on the second hop system (`lbl-csam.arpa`) that forwards packets with a 0 ttl (this is a bug in the distributed version of 4.3 BSD UNIX systems). You must guess the path that the packets are taking across the network, because the NSFNet (129.140) does not supply address-to-name translations for its NSSs. The two numbers at the end of each line are the median and mean times for the answers received from that host.

Example 2: A more interesting example is as follows.

```
[yak 72]% traceroute allspice.lcs.mit.edu.
traceroute to allspice.lcs.mit.edu (18.26.0.115), 30 hops max
 1  helios.ee.lbl.gov (128.3.112.1)  0 ms  0 ms  0 ms (0.0/0.000)
 2  lilac-dmc.Berkeley.EDU (128.32.216.1)  19 ms  19 ms  19 ms (19.0/19.0)
 3  lilac-dmc.Berkeley.EDU (128.32.216.1)  39 ms  19 ms  19 ms (19.0/25.667)
 4  ccngw-ner-cc.Berkeley.EDU (128.32.136.23)  19 ms  39 ms  39 ms (39.0/32.333)
 5  ccn-nerif22.Berkeley.EDU (128.32.168.22)  20 ms  39 ms  39 ms (39.0/32.667)
 6  128.32.197.4 (128.32.197.4)  59 ms  119 ms  39 ms (59/72.333)
 7  131.119.2.5 (131.119.2.5)  59 ms  59 ms  39 ms (59.0/52.333)
 8  129.140.70.13 (129.140.70.13)  80 ms  79 ms  99 ms (80.0/86.000)
 9  129.140.71.6 (129.140.71.6)  139 ms  139 ms  159 ms (139.0/145.667)
10  129.140.81.7 (129.140.81.7)  199 ms  180 ms  300 ms (199.0/226.333)
11  129.140.72.17 (129.140.72.17)  300 ms  239 ms  239 ms (239.0/259.333)
12  * * *
13  128.121.54.72 (128.121.54.72)  259 ms  499 ms  279 ms (279.0/345.667)
14  * * *
15  * * *
16  * * *
17  * * *
18  ALLSPICE.LCS.MIT.EDU (18.26.0.115)  339 ms  279 ms  279 ms (279.0/299.000)
```

The gateways that are 12, 14, 15, 16, and 17 hops away either do not send ICMP `time exceeded` messages or send them with a ttl too small to reach `traceroute`. Gateways 14 through 17 are running the Massachusetts Institute of Technology (MIT) C Gateway code that does not send `time exceeded` messages. No information is available for gateway 12.

The silent gateway 12 in example 2 may be the result of a bug in the 4.2 BSD or 4.3 BSD network code (and its derivatives); the 4.0, 4.1, 4.2, and 4.3 versions of UNIX send an unreachable message, using whatever ttl remains in the original datagram. Because, for gateways, the remaining ttl is 0, the ICMP `time exceeded` is guaranteed to not make it back to `traceroute`. The behavior of this bug is slightly more interesting when it appears on the destination system, as in example 3.

Example 3:

```
 1 helios.ee.lbl.gov (128.3.112.1)  0 ms  0 ms  0 ms (0.0/0.000)
 2 lilac-dmc.Berkeley.EDU (128.32.216.1)  39 ms  19 ms  39 ms (39.0/32.333)
 3 lilac-dmc.Berkeley.EDU (128.32.216.1)  19 ms  39 ms  19 ms (19.0/25.667)
 4 ccngw-ner-cc.Berkeley.EDU (128.32.136.23)  39 ms  40 ms  19 ms (39.0/32.667)
 5 ccn-nerif35.Berkeley.EDU (128.32.168.35)  39 ms  39 ms  39 ms (39.0/39.000)
 6 csgw.Berkeley.EDU (128.32.133.254)  39 ms  59 ms  39 ms (39.0/45.667)
 7 * * *
 8 * * *
 9 * * *
10 * * *
11 * * *
12 * * *
13 rip.Berkeley.EDU (128.32.131.22)  59 ms !  39 ms !  39 ms ! (39.0/45.667)
```

There are 12 gateways (13 is the final destination), and exactly the last half of them are missing. Actually, `rip` (a Sun-3 workstation running SunOS 3.5 system) is using the ttl from the arriving datagram as the ttl in its ICMP reply. Therefore, the reply times out on the return path (with no notice sent to anyone because ICMPs are not sent for ICMPs) until `traceroute` probes with a ttl that is at least twice the path length; that is, `rip` is really only 7 hops away. If a reply returns with a ttl of 1, this problem probably exists. If the ttl is ≤ 1, `traceroute` prints a `!` character after the time. Because vendors ship some obsolete (DEC ULTRIX, Sun 3.*x*) or nonstandard (HPUX) software, expect to see this problem frequently and/or take care when you select the target host of your probes.

Other possible annotations after the time are `!H`, `!N`, `!P` (host, network, or protocol unreachable, respectively), `!S`, or `!F` (source route failed or fragmentation needed); these should be seen only if you specified a source route, or the `-f` flag; otherwise, neither of these should occur, and the associated gateway is broken if you see one. If most probes result in some kind of unreachable message, `traceroute` exits.

SEE ALSO

`ping`(8)

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

NAME

`trcollect` – Collects trace information for TCP/IP, and NFS

SYNOPSIS

`/etc/trcollect` [`-b` *readbuffers*] [`-f` *tracefile*] [`-i` *interface*} [`-r` *recvspace*] [`-s` *timeout*] [`-t` *connection*] [`-u` *connection*] [*generic_info*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `trcollect` command is a trace collecting program for TCP/IP, and NFS protocols. It collects trace information by creating a trace socket and connecting to various entities in the protocol stacks. The options and arguments on the command line determine these entities. The following types of connections are available:

Generic	Traces all information through a specfic type.
Interface specific	Traces information through a specific interface.
Connection specific	Traces information through a particular TCP or UDP connection.

`trcollect` accepts the following options and operand:

`-b` *readbuffers* Specifies the size of the trace buffers in bytes; if the number is followed by a `k` or `K`, the size is in Kbytes. This is the amount of data that TCP/IP buffers before queuing it to satisfy the read request from `trcollect`. The default is 1024 bytes.

`-f` *tracefile* Specifies the name of the trace file to which `trcollect` writes the trace data. If `-f` is not specified, trace data is written to standard output; therefore, the same effect is accomplished by redirecting standard output to a file.

`-i` *interface* Specifies that interface-specific tracing be performed on the specified interface. For example, `-i hy0` traces data through interface `hy0`.

`-r` *recvspace* Specifies the receive space for the socket in bytes; if the number is followed by a `k` or `K`, the size is in Kbytes. This is the amount of data that TCP/IP buffers, while waiting for `trcollect` to issue a read, before dropping data. The default is 32 Kbytes. If the specified size exceeds the maximum for a socket, it can be truncated.

`-s` *timeout* Specifies a time-out period (in seconds) on read operations. This indicates the amount of time that `trcollect` waits for TCP/IP to fill one read buffer before putting data that has been received onto the receive queue. By default, this occurs every 5 seconds.

`-t` *connection* Specifies TCP connection-specific tracing. All TCP-related information for the specified connection is traced. The following four parameters, which must be surrounded by single or double quotation marks, define a connection:

`-fa` *foreign address*	Remote host's Internet address or name.
`-fp` *foreign port*	Remote host's port number or name, as given in `/etc/services` (see `services`(5)).
`-la` *local address*	Local host's Internet address or name.
`-lp` *local port*	Local host's port number or name, as given in `/etc/services`.

If you omit any one of these parameters, it will be used as a wildcard, that is, all of the types defined by the other three parameters are traced. TCP connection-specific tracing requires at least one option.

Examples:

To trace any TCP connection with foreign address *host1*, foreign port *port1*, local address *host2*, and local port *port2*, you should specify the following as the operand for the `-t` option:

```
-t '-fa host1 -fp port1 -lp port2 -la host2 '
```

To trace any TCP connection being established from port *port1* on foreign host *host1*, you should specify the following as the operand for the `-t` option:

```
-t '-fa host1 -fp port1 '
```

`-u` *connection* Specifies UDP connection-specific tracing. All UDP-related information for the specified connection is traced. The following four parameters, which must be surrounded by single or double quotation marks, define a connection:

`-fa` *foreign address*	Remote host's Internet address or name.
`-fp` *foreign port*	Remote host's port number or name, as given in `/etc/services` (see `services`(5)).
`-la` *local address*	Local host's Internet address or name.
`-lp` *local port*	Local host's port number or name, as given in `/etc/services`.

If you omit any one of these parameters, it will be used as a wildcard, that is, all of the types defined by the other three parameters are traced. UDP connection-specific tracing requires at least one option.

Examples:

To trace any UDP connection with foreign address *host1*, foreign port *port1*, local address *host2*, and local port *port2*, you should specify the following as the operand for the `-u` option:

```
-u '-fa host1 -fp port1 -lp port2 -la host2 '
```

To trace any UDP connection being established from port *port1* on foreign host *host1*, you should specify the following as the operand for the `-u` option:

```
-u '-fa host1 -fp port1 '
```

generic_info	Specifies the type of tracing information. The valid types and their meanings are as follows:

`icmp`	Specifies tracing information through the ICMP layer.
`idmap`	Specifies tracing information through the NFS `idmap` scheme.
`if`	Specifies tracing information on all interfaces.
`ip`	Specifies tracing information through the IP layer.
`mbuf`	Specifies tracing information on all mbuf allocations and deallocations. (This feature is not yet implemented.)
`nfs`	Specifies tracing information through NFS. Note: `trcollect` does not support the Cray-to-Cray NFS protocol, CNFS. See NOTES for more information.
`profile`	Specifies tracing information through the kernel subroutines, which include the driver interface subroutine, the socket send subroutine, and so on.
`rawip`	Specifies tracing information through the raw IP layer.
`rpc`	Specifies tracing information on all remote procedure call (RPC) requests. (Deferred implementation.)
`socket`	Specifies tracing information on all socket changes. (Deferred implementation.)
`tcp`	Specifies tracing information through all TCP sockets that have the `SO_DEBUG` flag set.
`udp`	Specifies tracing information through all UDP sockets that have the `SO_DEBUG` flag set.

NOTES

NFS read and write traces are frequently lost. To use the `nfs` option to trace NFS read and write requests, the `-b readbuffers` option must be specified with a size of at least 8192. The size should be the same as the value specified for `rsize` and `wsize` on the `mount`(8) command.

Also, it might be necessary to specify the `-r recvspace` parameter to trace NFS. The value of `recvspace` for the `-r` parameter should be a multiple of the size specified for the `-b` parameter. Increase the size of the `-r` parameter until NFS traces are no longer missing.

BUGS

The `-i` option is not supported on the EL series.

SEE ALSO

`trformat`(8)

NAME

`trformat` – Formats trace information obtained from `trcollect`(8)

SYNOPSIS

`/etc/trformat` [`-c`] [`-d`] [`-f` *tracefile*] [`-h`] [`-m` *hex_bitmap*] [`-n`] [`-s` *nfs_bitmap*] [`-t`] [`-v`] [`-x`] [*types*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `trformat` command is a trace formatting program for TCP/IP and NFS protocols. It formats the trace information collected by the trace collecting program, `trcollect`(8).

The `trformat` command accepts the following options and operand:

`-c`	Checks the sequence numbers in the entry to determine whether entries are missing.
`-d`	Prints the `ack` and `seq` fields in `tcp` entries in decimal format instead of hexadecimal format.
`-f` *tracefile*	Specifies the name of the trace file produced by `trcollect`. If you omit `-f`, trace data is read from standard input; therefore, the same effect is accomplished by redirecting the data file to standard input.
`-h`	(Header) Displays only the trace entry header information.
`-m` *hex_bitmap*	Specifies the types of ID map entries to be formatted. `hex_bitmask` is a hexadecimal number that uses a value of `1` in a bit position if that type will be formatted and uses a value of `0` if that type will be skipped. The default is to format all entries.
`-n`	Displays addresses in dot format and ports as integers.
`-s` *nfs_bitmap*	Specifies the types of NFS entries to be formatted. `nfs_bitmap` is a hexadecimal number that uses a value of `1` in a bit position if that type will be formatted and uses a value of `0` if that type will be skipped. The default is to format all entries.
`-t`	In the `rtc` field, prints the elapsed time (in milleseconds) instead of the real-time clock value.
`-v`	Verbose mode. Displays detailed information for each entry.
`-x`	(Hexidecimal) Displays each trace entry in hexadecimal format.
types	Displays only the entries of the specified type or types. You can use this operand to filter the necessary entries from a trace file. The valid types and their meanings are as follows:

`icmp`	Displays trace information obtained through the ICMP layer.
`idmap`	Displays trace information obtained through the NFS `idmap` scheme.

`if`	Displays trace information obtained from all interfaces.
`ip`	Displays trace information obtained through the IP layer.
`mbuf`	Displays trace information obtained on all mbuf allocations and deallocations. (Deferred implementation.)
`nfs`	Displays trace information obtained through NFS.
`profile`	Displays trace information obtained through the kernel subroutines. These include the driver interface subroutine, the socket send subroutine, and so on.
`rawip`	Displays trace information obtained through the raw IP layer.
`rpc`	Displays trace information obtained on all RPC requests. (Deferred implementation.)
`socket`	Displays trace information obtained on all socket changes. (Deferred implementation.)
`tcp`	Displays trace information obtained on all TCP sockets with the `SO_DEBUG` flag set.
`udp`	Displays trace information obtained on all UDP sockets with the `SO_DEBUG` flag set.

SEE ALSO

`trcollect`(8)

NAME

`trpt` – Transliterates protocol trace

SYNOPSIS

`/etc/trpt` [`-a`] [`-f`] [`-j`] [`-k`] [`-p` *hex-address*] [`-s`] [`-t`] [*system* [*core*]]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `trpt` procedure interrogates the buffer of Transmission Control Protocol (TCP) trace records created when a socket is marked for debugging (see `setsockopt`(2)) and prints a readable description of these records. When you omit all arguments, `trpt` prints all of the trace records found in the system grouped according to TCP connection Protocol Control Block (PCB). `trpt` accepts the following options to alter this behavior:

`-a` In addition to the normal output, prints the values of the source and destination addresses for each packet recorded.

`-f` Follows the trace as it occurs, waiting a short time for additional records each time the end of the log is reached.

`-j` Gives a list of only the those PCB addresses for which trace records exist.

`-k` The `-k` option is provided for debugging purposes. It forces `trpt` to use `/dev/kmem` to obtain data, rather than use system calls.

`-p` *hex-address*
Shows only trace records associated with the PCB whose address follows the block.

`-s` In addition to the normal output, prints a detailed description of the packet sequencing information.

`-t` In addition to the normal output, prints the values for all timers at each point in the trace.

system [*core*]
Allows you to specify alternate kernel and core dump files. The default is `/unicos` and `/dev/kmem`. These arguments allow `trpt` to be used on a core dump.

The recommended use of `trpt` is as follows:

1. Isolate the problem and enable debugging on the sockets involved in the connection.
2. Find the address of the PCBs associated with the sockets by using the `-A` option to `netstat`(1B).
3. Run `trpt` with the `-p` option, supplying the associated PCB addresses. You can use the `-f` option to follow the trace log after the trace is located. If many sockets are using the debugging argument, the `-j` option can be useful in checking to see whether any trace records are present for the socket in question.

MESSAGES

`no namelist`	The system image does not contain the proper symbols to find the trace buffer.

BUGS

The `trpt` procedure should also print the data for each input or output, but this is not saved in the `trace` record.

The output format is inscrutable and should be described.

FILES

`/dev/kmem`	Kernel memory
`/unicos`	Default core dump file

SEE ALSO

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

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

NAME

`tsar` – System data processing language

SYNOPSIS

`/usr/bin/tsar` [`-c`] [`-d` *flags*] [`-v`] [*source_files*]

`/usr/bin/tsar` [`-d` *flags*] [`-e` *end_time*] [`-p` *playback_file*] [`-r` *record_file*] [`-s` *start_time*] [`-v`] [`-D` *name*[=*def*]] [*source_files*]

`/usr/bin/tsar` [`-d` *flags*] [`-e` *end_time*] [`-h` *host* [`-i` *interval*] [`-n` *sample_count*] [`-r` *record_file*] [`-P` *path*]] [`-s` *start_time*] [`-v`] [`-D` *name*[=*def*]] [*source_files*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `tsar` command reads system activity data records that have been produced on a UNICOS system by the `sdc`(8) command. By default, the programs are taken from `stdin`. `tsar` uses programs written in a language based on `nawk` to process the data. The *source_files* operand is a command file which specifies the names of these programs.

The `tsar` command operates in one of three modes: compilation_only, playback, or online. Use the `-c`, `-p`, and `-h` options to enter the respective modes. compilation_only mode is used to debug source files. In playback mode, `tsar` processes data from an existing system activity data file (*playback_file*). In the online mode, `tsar` processes the data as it is generated on the host system by the `sdc`(8) command. By default, `tsar` executes in playback mode.

The `tsar` language is described in detail in *UNICOS Resource Administration*, Cray Research publication SG–2302.

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

`-c`	Compiles only the source files. Use this option to debug source files.

`-d` *flags*	Specifies the debug flags. The flags are as follows: 0001 Lexical scanning 0002 Expression compilation 0004 Table entry 0010 Code execution 0020 Stack contents 0040 Playback file parsing 0100 Symbol table searching 0200 Table allocation Multiple flags can be specified by adding the numerical values. For example, to enable lexical scanning and code execution debugging, set the flag to 011. Debug output is written to `stderr`.
`-e` *end_time*	Specifies the end time of the report, using the form *hh*[:*mm*[:*ss*]]. This option can be used only in playback mode.
`-h` *host*	Specifies the name of the remote Cray Research host system where data sampling is to occur, and places `tsar` in the online mode.
`-i` *interval*	Specifies the data sampling interval. *interval* is of the form *xx*`h`*yy*`m`*zz*`s` (*xx* hours, *yy* minutes, *zz* seconds) or *zz* seconds. By default, data is sampled every 5 minutes. This option can be used only in online mode.
`-n` *sample_count*	Specifies the number of samples to be taken when `tsar` is in the online mode. By default, `tsar` collects five data samples.
`-p` *playback_file*	Specifies the file from which `tsar` extracts data. *playback_file* is the output from `sdc`(8). Use of this option places `tsar` in playback mode. The default playback file is the `dcf` file in the current directory.
`-r` *record_file*	Specifies the file where the data samples are to be recorded. In online mode, `sdc`(8) writes its output to *record_file*. In playback mode, *record_file* will be a copy of the playback file.
`-s` *start_time*	Specifies the start time of the report. The time is specified using the form *hh*[:*mm*[:*ss*]]. This option can be used only in playback mode.
`-v`	Specifies verbose output when `tsar` processes a source file. Output is written to `stderr`.
`-D` *name*[=*def*]	Defines a symbol *name* to be used in the source file during execution. *def* can be a number or a character string. Character strings must be delimited by escaped double quotation marks. By default, *def* is defined as the number 1.
`-P` *path*	Specifies the path where the `sdc`(8) binary is to be found. This option can be used only with the `-h` option. By default, if `sh` or `ksh` is used, the `PATH` in `/etc/profile` is searched. If `csh` is used, the path in your `.cshrc` file is searched.
source_files	A command file which specifies the names of `tsar` programs.

NOTES

When using the `-h` option, ensure that the appropriate `.rhosts` file on the remote host is set up properly. Otherwise, `sdc`(8) may abort with a `Permission denied` error.

If more than one *source_file* is specified, `tsar` executes the files in sequence. The preambles, bodies, and postambles are processed as if they came from a single source file.

EXAMPLES

Example 1: The following example shows `tsar` being executed in online mode. `tsar` collects ten data samples at a rate of one sample every 5 minutes. The data is collected from a machine called `haze`, and data is written to the file `dcf`. As the data is collected, it is formatted according to the source file `mem.ts`. The formatted output is written to `stdout`.

```
$ tsar -h haze -n 10 -i 5m -r dcf mem.ts
```

Example 2: The following example shows `tsar` being executed in playback mode. The data in the file `dcf` is formatted according to the source file `rpt.ts`. The symbol `CPU_RPT` is defined as 2.

```
$ tsar -D CPU_RPT=2 rpt.ts
      or
$ tsar -p dcf -D CPU_RPT=2 rpt.ts
```

SEE ALSO

`sdc`(8)

UNICOS Resource Administration, Cray Research publication SG–2302

A. V. Aho, B. W. Kernighan, P. J. Weinberger, *The AWK Programming Language*, Addison-Wesley, 1988.

NAME

`turnacct` – Controls process accounting

SYNOPSIS

```
/usr/lib/acct/turnacct on
/usr/lib/acct/turnacct off
/usr/lib/acct/turnacct switch
```

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `turnacct` shell script is an interface to the `accton`(8) command to turn process accounting on or off or to switch files for maintainability. A super user or a user who is in the group, `adm`, and has permission bit `acct` set in their user database (UDB) entry (see `udbgen`(8)) must invoke `turnacct`.

The `turnacct` shell script accepts the following mutually exclusive operands:

`on`	Turns on process accounting. If the process accounting file `/usr/adm/acct/day/pacct` exists, accounting records are appended to it; otherwise, `turnacct` creates a new file and accounting records are written to it.
`off`	Turns off process accounting.
`switch`	Specifies that a new process accounting file should be created to maintain manageable files. `turnacct` moves the current process accounting file `/usr/adm/acct/day/pacct` to the next available file `/usr/adm/acct/day/pacct*` and then calls `accton` by using the file operand `/usr/adm/acct/day/pacct`. In this way, process accounting data is not lost and `/usr/adm/acct/day/pacct` remains the current file.

NOTES

The shell script, `ckpacct`(8), invokes `turnacct` to maintain process accounting files of a reasonable size. You should run `ckpacct`(8) periodically by using `cron`(8).

If the super user has run any accounting commands or shell scripts, you may set up the group ID and permissions of the accounting files so that any user in the group, `adm`, with the permission bit `acct` set in his or her UDB entry cannot run `accton`. To change all of the accounting file's group ID and permissions as necessary, use the `csaperm`(8) command.

FILES

`/usr/adm/acct/day/pacct`	Current process accounting file
`/usr/adm/acct/day/pacct*`	Switched process accounting files

SEE ALSO

`acct`(8), `accton`(8), `acctsh`(8), `ckpacct`(8), `cron`(8), `csa`(8), `csaperm`(8), `udbgen`(8)

UNICOS Resource Administration, Cray Research publication SG–2302

NAME

`turndacct` – Controls daemon accounting

SYNOPSIS

`/usr/lib/acct/turndacct on` *daemon*
`/usr/lib/acct/turndacct off` *daemon*
`/usr/lib/acct/turndacct switch` *daemon*

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `turndacct` shell script is an interface to the `csaswitch` command (see `csaswitch`(8)) to turn daemon accounting on or off or to switch files for maintainability. A super user or a user who is in the group, `adm`, and has permission bit `acct` set in their user database (UDB) entry (see `udbgen`(8)) must invoke `turndacct`.

The `turndacct` shell script accepts the following mutually exclusive operands:

`on` Turns on daemon accounting for the specified *daemons*.

`off` Turns off daemon accounting for the specified *daemons*.

`switch` Specifies that a new daemon accounting file should be created for each of the specified daemons to maintain manageable files. `turndacct` moves the current daemon accounting file `/usr/adm/acct/day/daemonacct` to the next available file, `/usr/adm/acct/day/daemonacct#`, and then calls `csaswitch` by using the file operand `/usr/adm/acct/day/daemonacct`. In this way, daemon accounting data is not lost, and `/usr/adm/acct/day/daemonacct` remains the current file.

daemon Valid daemon names are `nqs`, `tape` and `socket`. This operand is required in each of the three command formats described previously.

NOTES

The `ckdacct`(8) shell script invokes `turndacct` to maintain process accounting files of a reasonable size. You should run `ckdacct` periodically by using `cron`(8).

If the super user has run any accounting commands or shell scripts, you may set up the group ID and permissions of the accounting files so that any user in the group, `adm`, with the permission bit `acct` set in his or her UDB entry cannot run `accton`. To change all of the accounting file's group ID and permissions as necessary, use the `csaperm`(8) command.

EXAMPLES

The following example turns off Network Queuing System (NQS) accounting:

```
turndacct off nqs
```

FILES

`/usr/adm/acct/day`	Directory that contains current daemon accounting files

SEE ALSO

`acct`(8), `accton`(8), `acctsh`(8), `ckdacct`(8), `cron`(8), `csa`(8), `udbgen`(8)

UNICOS Resource Administration, Cray Research publication SG–2302

NAME

`udbgen` – Generates or maintains the user database

SYNOPSIS

`/etc/udbgen` [`-n`] [`-q`] [`-u`] [`-v`] [`-p` *UDB_path*] [`-t` *type*] [*infile*]
`/etc/udbgen` [`-n`] [`-q`] [`-u`] [`-v`] [`-p` *UDB_path*] [`-c` *command*]
`/etc/udbgen -s` [`-q`] [`-u`] [`-p` *UDB_path*]
`/etc/udbgen -a` [`-A`] [`-R`] [`-p` *UDB_path*]
`/etc/udbgen -h`
`/etc/udbgen -m`

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `udbgen` command creates and maintains information in the user database (UDB). The UDB contains comprehensive information about each user and is the primary source of that information in the UNICOS operating system. For more information on the UDB, see the `udb`(5) man page.

The `udbgen` command is intended for use by administrators. The UNICOS system may enforce further restrictions on various types of administrators concerning who may change sensitive fields in the UDB.

After updating the UDB, `udbgen` enters the associated file maintenance phase and updates `/etc/acid` files, `/etc/group` files, and `/etc/passwd` files, because some commands assume that they exist. (See the `acid`(5), `group`(5), and `passwd`(5) man pages for more information on these files.) If an account or group ID appears in the UDB without being specified in the existing `/etc/acid` or `/etc/group` files, a name is created for it by appending the ID number to the prefix `A-` (for `/etc/acid`) or `G-` (for `/etc/group`). The ID suffix is a minimum of five digits, such as `A-00921`.

A new `/etc/passwd` file, created entirely from the UDB, replaces the existing `/etc/passwd` file.

The number of names created are displayed in the statistics output if the `-q` option has not been selected.

There are three possible sources for the information `udbgen` places in the UDB, as follows:

- The input file, *infile*. This file can be either in UDB source format, in the form of the `/etc/passwd` file, or in the form of the `/etc/uentry` file. UDB source format is discussed in the Creating Input for `udbgen` subsection. When reading `/etc/passwd` or `/etc/uentry` type source, `udbgen` either creates or updates entries. If the user name already exists in the UDB, the name is updated; otherwise, the name is created. This decision is made on every line of input.

  WARNING: Support for `/etc/passwd` and `/etc/uentry` formats is included only for upgrading from previous releases of UNICOS (before release 5.0); do not use these formats for any other purpose.

- A directive statement used with the `-c` option. This statement must be in UDB source format.

- Statements entered interactively. `udbgen` enters the interactive mode if an input file is not specified on the command line and `stdin` is connected to a terminal.

The `udbgen` command accepts the following options and operand:

`-a` Analyzes the fair-share resource assignments in the UDB (using the `shrtree`(8) utility) and reports any problems. A nonzero exit status occurs if any potential problems are found. This report is for use by the administrator as determined by site policy; no changes to the UDB are made. Only the `-A`, `-p`, or `-R` options can be used with the `-a` option.

`-A` In conjunction with the `-a` option, this option causes fair-share resource group analysis to be based on the `acids` field rather than the `resgrp` field in the UDB. This is to be used if the fair-share scheduler is run in its Share by Account mode. The `-A` option can be used only with the `-a` option and not in combination with the `-R` option.

`-c` *command*
Specifies command-line input. This option allows simple database changes to be made without an input file. The *command* argument is a UDB source format directive or set of directives. When this option is used, the `-q` option is automatically invoked and no input file is read (specifying *infile* is an error when this option is specified). For example, the following command line sets the shell for user `xyz` in the database to `/bin/csh`:

```
udbgen -c 'update:xyz: shell:/bin/csh:'
```

For a list of available directives and possible field values, see the Creating Input for `udbgen` subsection.

`-h` Displays the command syntax and quits. All other options are ignored.

`-m` Resets the UDB maximum user ID (UID) value to the highest non-deleted UID. The UDB maintains a record of the highest UID value ever assigned. This value is then used to determine the UID when creating a new record using the `:uid:next:` syntax. Since this value is never decremented, even when records are deleted, it is possible to reach large UID values while leaving large ranges of UIDs unused. This option allows you to reset this value without having to rebuild the entire UDB from source.

All other options are ignored.

NOTE: Take care when using this option. For example, if the UDB record with the highest UID is deleted and then the `-m` option is used to reset the maximum UID value, the deleted UID value might be assigned to another user. If the original user had any files remaining on the system, they would come under the ownership of the newly created user.

`-n` Adds new user names only and does not affect existing records. Without this option, `udbgen` adds new user names and creates, deletes, and updates existing records in the UDB.

`-p` *UDB_path*
Sets the path name for access to the UDB files. The default path name is `/etc`. This option allows private or test versions of the UDB files to be created and maintained. The path name must end with a directory name. For example, if you want to specify that the UDB files are to be in the `/c/abc/UDBtest` directory, you would enter the following command line:

```
udbgen -p /c/abc/UDBtest
```

If you use the `-p` option and you are not appropriately authorized, `udbgen` eliminates all of its special abilities and runs with the normal abilities of the user. This behavior means that such users are constrained by the normal access controls of UNICOS.

The path names for files in the FILES section can be changed with this option.

`-q` Specifies quiet mode. This option prevents informative messages from appearing.

`-R` In conjunction with the `-a` option, this option causes fair-share resource group analysis to be based on the `resgrp` field in the UDB. This is to be used if the fair-share scheduler is run in its Share by User mode. This is the default mode if the `-a` option appears alone. The `-R` option can be used only with the `-a` option and not in combination with the `-A` option.

`-s` Updates `/etc/acid`, `/etc/group`, and `/etc/passwd` files to accurately reflect the contents of the UDB. Only the `-p`, `-q`, and `-u` options can be used with `-s`.

`-t` *type*
Specifies the format of the data in *infile*. The *type* argument may be one of the following values:

Value	**Description**
`passwd`	Password format. This type is provided only for upgrading from previous releases of UNICOS (before release 5.0); do not use this type for any other purpose, because it accommodates only a small part of the information contained in the UDB.
`uentry`	`uentry` format. This type is provided only for upgrading from previous releases of UNICOS (before release 5.0); do not use this type for any other purpose, because it accommodates only a small part of the information contained in the UDB.
`udb`	UDB format.

Ordinarily it is not necessary to specify the input format because `udbgen` analyzes the first line of the file to determine which format is in use. This option bypasses this analysis and must be used if the following message is displayed:

```
udbgen cannot determine input style
```

If *infile* is associated with a terminal device, this option is ignored and *type* is set to UDB format.

`-u` Reports database access statistics.

`-v` Verifies the accuracy of the input file only. The database is not updated with this option.

infile Name of the source data file. The default is `stdin`. If the file is associated with a terminal device, `udbgen` enters interactive mode and displays a prompt (`>`) to `stdout` before every read. The end of interactive input must be specified by a new line that contains `quit` or `q`.

The `udbsee`(1) command can be used to produce a file suitable as a source data file.

Creating Input for udbgen

A source data file (or directive used with the `-c` option or used interactively) contains the following information:

- Block introduction field name, usually consisting of a command to `udbgen` and a user name, separated by colons. See the Block Introduction Field Names subsection.
- Field names other than the user name, sometimes with editing suffixes, separated by colons. See the Field Names and Values subsection.
- Field values. See the Field Names and Values subsection.

To create an input file for `udbgen` to use in updating the UDB, you must follow these file format rules:

- Comments can be placed in the source code by using the character # (similar to shell script comments). Comments continue from the # to the end of the line on which the # appears. Comments are not recognized within colon-delimited fields. Blank lines are allowed in the source code.
- White space (blanks, tabs, and so on, as determined by `isspace` (see `ctype`(3C)) may appear freely in the source, but it is removed during parsing, except within some colon-delimited fields where white space is part of the field value. (The Field Names and Values subsection states where white space is retained in fields.)
- Each default block must begin on a new line with the field name `default`. (See the Block Introduction Field Names subsection.)
- Each user description block must begin on a new line with one of the field names `create`, `delete`, or `update`. This is how one user description is separated from the next. A user ID must be specified when `create` is used. (See the Block Introduction Field Names subsection.)
- Each field name must be followed by a colon-delimited field, according to the following format:

  field_name :*field_value*:

  This name/value pairing is implicit in the source definition and must, but for one exception, be followed exactly for the input file to be accepted. The exception occurs with the special word `quit` or `q`, which is recognized only in interactive mode as an input terminator.
- Fields that have batch and interactive components must have the appropriate indicator (`[b]` or `[i]`) following the field name (separating white space is optional). For example, the following line sets the interactive tape limit of tape type 0 to 1.

  ```
  jtapelim[i][0] :1:
  ```

Block Introduction Field Names

The special block introduction field names update the UDB and are used in the input file, with the `-c` option as directive statements, and with `udbgen` interactively. The field names consist of a command to `udbgen` and a user name, separated by colons. All field names must be entered in lowercase in the source file. Fields not included in a user description are set to an appropriate default value. Strings (except for `passwd`) are null, fields that have a default value are set to that value (for a description of `default`, see the Global Default Table subsection), and other numeric and bit fields are set to `0`. An unspecified list of accounts (`acids`) or groups (`gids`) means that every entry in the associated array is set to `-1`. An undeclared password is set to the string `*`.

`create :`*user_name*`:uid :`*n*`:`
: Adds the specified user's definition to the UDB. This is one of the recognized initial reserved words for a definition block and must appear first. If a record of this name is already present, a warning message is displayed and the existing record is not changed. The user ID must be specified with the `create` command.

  User names are limited by the UNICOS operating system to 8 characters; `udbgen` issues a warning if this restriction is violated.

`delete :`*user_name*`:`
: Removes the specified user's definition from the UDB. This is one of the recognized initial reserved words for a definition block and must appear first. While other information may be present in this definition block, it is ignored. A warning message is displayed if the specified user name does not appear in the database.

`global :default:`
: Changes the values of the default fields named in the block. For a description of `default`, see the Global Default Table subsection.

`global :tmap:`
: Changes the names of the eight tape types defined in the global tape map. For a description of `tmap`, see the Global Tape Name Map subsection.

`quit`
`q`
: Simulates the end of a file. This field name is recognized only in interactive mode (see the previous description of *infile*) and is recognized only if it is alone on a line with no extraneous white space. This is the single exception to the pairing rule.

`see :`*user_name*`:`
`see` *user_name*
: Displays a UDB entry by executing the `udbsee`(1) command. This field name is recognized only in interactive mode (see the previous description of *infile*) and must be alone on a line. This action does not disturb any ongoing input to `udbgen`. This field accepts either the colon delimiters or white space separating the field name from the field value.

`update :`*user_name*`:`

Updates fields defined in this block in the UDB. The user name must exist in the UDB. This is one of the recognized initial reserved words for a definition block and must appear first. Fields not mentioned retain their original values. Fields that are either bit or numeric arrays have the values supplied in an `update` block processed as directed by the editing suffix immediately following the field name. For a detailed explanation of the editing suffixes, see the Field Names and Values subsection.

If, during an `update`, the user ID is changed, the record is first deleted from the database and then added with the new user ID. No checking is done to ensure that no other user is assigned to the new user ID.

Field Names and Values

Field names describe various types of data contained in the UDB, including user limits, quotas, and privileges. Editing suffixes immediately follow the field name. The editing suffix = may be added to a field name to indicate that the value following is the value to be entered in the UDB. The = suffix is optional in all cases. The suffixes + and - allow you to update a field value that is either bit or numeric arrays using the `update` block. Field editing is allowed only with `update`.

In `update` mode, the numeric arrays can be altered with the editing suffixes + or -. Assuming a field is specified more than once in a definition block, processing is done as follows:

1. If replacement values are provided, they are entered in the array; otherwise, the values from the existing record are used.
2. All values appearing with a + suffix are appended to the array.
3. Any duplicates are deleted.
4. All values appearing with a - suffix that exist in the array are removed.
5. The array is packed to move all empty space to the end. At no time may the result array exceed its limit of entries; therefore, in full or nearly full arrays it is possible to get space-exceeded warnings even when the final result of the editing would not exceed the maximum array size. To avoid this, you can split the `update` block into two separate operations and then delete before appending, as in the following example:

   ```
   update :name: acids- :23,45,87: acids+ :1,2,3,4:
   ```

   To remove all values from an array (such as `acids`), use the following method:

   ```
   update :name: acids= :1: acids- :1:
   ```

   The value of `1` is arbitrary; any legal value may be used.

For example, if, for the user `anne`, the `acids` field contains the values 2, 34, and 45, you can remove 34 from the list by entering the following command line:

```
update :anne: acids-:34:
```

To add `97` to the list of account IDs, enter the following command line:

```
    update :anne: acids+:97:
```

To change the account ID list to `1`, `3`, and `5`, enter one of the following command lines:

```
    update :anne: acids:1,3,5:
    update :anne: acids=:1,3,5:
```

Field values are alphabetic and numeric, depending on the field. Numeric values are assumed to be decimal unless specifically noted as octal. Negative values are not allowed in any numeric field. Certain numeric fields also allow a special alphabetic notation for maximum values (all limits fields are in this category). This notation uses the words `none` or `unlimited` in lowercase or uppercase. (The words `infinite` and `infinity` in lowercase or uppercase are also supported, for compatibility reasons.) The appropriate numeric value is then stored in the UDB.

For example, to set an unlimited batch job CPU limit, enter the following command line:

```
    jcpulim[b] :unlimited:
```

To prohibit interactive SDS usage, enter the following command line:

```
    jsdslim[i] :none:
```

Internally, `none` results in a value of 0 and `unlimited` is a large value dependent on the data type of the field.

The following list contains field name / field value pairs, their descriptions, and possible values:

`acids =|+|- :`*n1*`,` *n2*`,` *...*`,` *nn*`:`
`acids =|+|- :`*a1*`,` *a2*`,` *...*`,` *an*`:`
Account IDs. This is a list of up to 64 (set by MAXVIDS) numeric account IDs or account names separated by commas. If account names are used, they must be found in the `/etc/acid` file as it existed before `udbgen` was executed.

`age :`*max*`,` *min*`:`
`age :force:`
`age :superuser:` This field name is obsolete, but it continues to be supported for compatibility reasons. For a description of its replacement, see the `pwage` field. Only an appropriately authorized user can change this field.

`archlim :`*n*`:` The maximum amount of the user's disk space protected from data migration. Users can protect files from migration by listing them in their `.keep` file. The files so listed are protected up to the number of disk blocks specified here.

`archmed :`*n*`:` Archive media selector for data migration. The value of this field determines to which media migrated files owned by the user are written. The media associated with the value 0 is the default media. The table of media is established in the DMF run-time configuration file. For complete information on establishing archive media, see "Data Migration" in the *Cray Data Migration Facility (DMF) Administrator's Guide*, Cray Research publication SG–2135.

`batchhost :`*text*`:` Host name for the last batch request. Up to 31 characters can be specified; white space is not removed.

`batchtime :`*seconds*`:`
Time of last batch request (GMT seconds).

`comment :`*text*`:` A comment consisting of a maximum of 39 characters; white space is not removed.

`comparts =|+|- :`*name1*`,` *name2*`,` *...*`,` *namen*`:`
Valid security compartment names. Only an appropriately authorized user can change this field.

`cpasswd :`*password*`:`
The clear text password to be encrypted and stored in the user's record. The password content is not validated. Only an appropriately authorized user can change this field.

NOTE: When creating a new account, see the `pwage` field. (Setting the `pwage` field to `force` ensures that the user will be required to enter a new password.)

`cpuquota :`*vv.v*`:` User CPU quota in seconds. If this field is nonzero, CPU quota is enforced to the value *vv.v* seconds. This value is stored internally in tenths of seconds but is expressed externally in seconds.

`cpuquotaused :`*vv.v*`:`
Amount of CPU quota used in seconds. This value is stored internally in tenths of seconds but is expressed externally in seconds.

NOTE: If this field is changed while the system is in multiuser mode, execute the command `shrsync -q` to synchronize the information in active lnodes with the changes in the UDB. For more information, see `shrsync`(8).

`defcomps =|+|- :`*name1*`,` *name2*`,` *...*`,` *namen*`:`
Default security compartment names. Only an appropriately authorized user can change this field.

`deflvl :`*n*`:` Default security level. Only an appropriately authorized user can change this field.

`dir :`*directory*`:` Default login directory consisting of a string of up to 63 characters.

`disabled :`*n*`:` User disabled indicator. If *n* is `0`, the user is enabled; if *n* is `1`, the user is disabled. Only an appropriately authorized user can change this field.

`gids =|+|- :`*n1*`,` *n2*`,` *...*`,` *nn*`:`
`gids =|+|- :`*g1*`,` *g2*`,` *...*`,` *gn*`:`
Group IDs. This is a list of up to 64 (set by MAXVIDS) numeric group IDs (GIDs) or group names separated by commas. If group names are used, they must be found in the `/etc/group` file as it existed before `udbgen` was executed. A warning message is issued if no GIDs are defined in a record. The UNICOS operating system is not intended to be operated without a GID for each user, but this is not a fatal error in order to be compatible with previous systems. This is especially important if the site uses network information systems (NIS) (formerly called yellow pages) software, because NIS does not work if a user does not have a GID.

`intcat =|+|- :`*name1*`,` *name2*`,` *...*`,` *nameN*`:`
Default categories. Only an appropriately authorized user can change this field.

`intcls :`*n*`:` Default class. Only an appropriately authorized user can change this field. This field is obsolete.

`jcpulim[b] :`*n*`:`
`jcpulim[i] :`*n*`:` Job CPU time limit, in seconds, for batch (`[b]`) or interactive (`[i]`) jobs. If *n* is `unlimited`, no restriction on the amount of CPU usage is applied. If this field is not specified, the appropriate value from the default table is used. The release default is `unlimited`.

`jfilelim[b] :`*n*`:`
`jfilelim[i] :`*n*`:` Per-job file allocation limit in 512-word blocks for batch (`[b]`) or interactive (`[i]`) jobs. If *n* is `zero` (default), there is no limit.

`jmemlim[b] :`*n*`:`
`jmemlim[i] :`*n*`:` Job memory limit, in 512-word blocks, for batch (`[b]`) or interactive (`[i]`) jobs. If *n* is `unlimited`, no restriction on the amount of memory usage is applied. If this field is not specified, the appropriate value from the default table is used. The release default is `unlimited`.

`jmppbarrier[b] :`*n*`:`
`jmppbarrier[i] :`*n*`:`
NOTE: Not implemented. Job massively parallel processing (MPP) barrier limit for batch (`[b]`) or interactive (`[i]`) jobs. (This field is present but not functional for Cray Research systems without a Cray MPP system.) If *n* is `unlimited`, no restriction on the number of barriers is applied. If this field is not specified, the appropriate value from the default table is used. The release default is `none`.

`jmpptime[b] :`*n*`:`
`jmpptime[i] :`*n*`:` Job MPP maximum reservation time limit, in wall-clock seconds, for batch (`[b]`) or interactive (`[i]`) jobs. (This field is present but not functional for Cray Research systems without a Cray MPP system.) If *n* is `unlimited`, no restriction on the MPP reservation time is applied. If this field is not specified, the appropriate value from the default table is used. The release default is `none`.

`jpelimit[b] :`*n*`:`
`jpelimit[i] :`*n*`:` Job MPP processing elements (PE) limit, for batch (`[b]`) or interactive (`[i]`) jobs. (This field is present but not functional for Cray Research systems without a Cray MPP system.) If *n* is `unlimited`, no restriction on the number of PEs is applied. If this field is not specified, the appropriate value from the default table is used. The release default is `none`.

`jproclim[b] :`*n*`:`
`jproclim[i] :`*n*`:` Job process limit for batch (`[b]`) or interactive (`[i]`) jobs. If this field is not specified, the default table entry will be used. The default table value for `jproclim` of the appropriate run class is either the release default returned from `sysconf(_SC_CHILD_MAX)` or the previously specified default table value. Setting *n* to `zero` or `none` indicates unlimited.

`jsdslim[b] :`*n*`:`
`jsdslim[i] :`*n*`:` Job secondary data segment limit, in 512-word blocks, for batch (`[b]`) or interactive (`[i]`) jobs. (This field is present but not functional for Cray Research systems without an SSD.) If *n* is `unlimited`, no restriction on the amount of secondary data segment usage is applied. If this field is not specified, the appropriate value from the default table is used. The release default is `none`.

`jshmsegs[b] :`*n*`:`
`jshmsegs[i] :`*n*`:` Maximum number of created shared memory segments for batch (`[b]`) or interactive (`[i]`) jobs. This field is always present, but is functional only for CRAY T90 systems supporting shared memory. If *n* is `unlimited`, the number of segments is limited by a configured upper limit (see `sys/config.h`). If *n* is `none`, shared memory cannot be used.

`jshmsize[b] :`*n*`:`
`jshmsize[i] :`*n*`:` Total shared memory segment size, in 512-word blocks, for batch (`[b]`) or interactive (`[i]`) jobs. This field is always present, but is functional only for CRAY T90 systems supporting shared memory. If *n* is `unlimited`, the number of segments is limited by a configured upper limit (see `sys/config.h`). If *n* is `none`, shared memory cannot be used.

`jsocbflim[b] :`*n*`:`
`jsocbflim[i] :`*n*`:` Total socket buffer space, in 512-word blocks, for batch (`[b]`) or interactive (`[i]`) jobs. If *n* is set to 0, unlimited use is allowed. (This is the default in the released system.)

`jtapelim[b]`[*t*] `:`*n*`:`
`jtapelim[i]`[*t*] `:`*n*`:`
Job tape unit limit for batch (`[b]`) or interactive (`[i]`) jobs. The value of *t*, the tape type, can be an integer in the range 0 through 7 or one of the names defined in the global tape name map. (For more information, see the Global Tape Name Map subsection.) By convention and for compatibility with previous releases, the value 0 corresponds to round tape types (`TAPE`) and the value 1 corresponds to cartridge tape types (`CART`). No names are defined by default in `tmap`. The maximum value of *n* is `unlimited` (254 for this type) and the released value is `none`. If this field is unspecified, the release default or the value previously specified in the default table will be used.

`limflags =|+|- :`*octal*`:`
`limflags =|+|- :`*name*, *name*, *...*`:`
This field name is obsolete, but it continues to be supported for compatibility reasons.

`logfails :`*n*`:` Number of consecutive login attempt failures since the last successful attempt. Only an appropriately authorized user can change this field.

`loghost :`*text*`:` Host name for the last login. Up to 31 characters can be specified, and white space is not removed.

`logline :`*text*`:` Line name used for last login. Up to 15 characters can be specified, and white space is not removed.

`logtime :`*seconds*`:` Time of last login (GMT seconds).

`maxcls :`*n*`:` Maximum class. Only an appropriately authorized user can change this field. This field is obsolete.

`maxlvl :`*n*`:` Maximum security level. Only an appropriately authorized user can change this field.

`minlvl :`*n*`:` Minimum security level. *n* may be any integer from 0 through 16; the default is 0. The value of `minlvl` should be less than, or equal to, the value of `maxlvl`. Only an appropriately authorized user can change this field.

`mincomps =|+|- :`*name1*, *name2*, *...*, *namen*`:`
Minimum security compartment names. Only an appropriately authorized user can change this field.

`nice[b] :`*n*`:`
`nice[i] :`*n*`:` Nice bias in the range $0 \le n \le 19$ for batch (`[b]`) or interactive (`[i]`) processes. If this field is not specified, the value from the default table or the released default value of 0 is used.

`parentuid :`*n*`:` UID of the group administrator for this entry.

`passwd :`*encrypted_password*`:`
The encrypted password to be stored in the user's record. The password content is not validated. Only an appropriately authorized user can change this field.

NOTE: When creating a new account, see the `pwage` field. (Setting the `pwage` field to `force` ensures that the user will be required to enter a new password.)

`pcorelim[b] :`*n*`:`
`pcorelim[i] :`*n*`:` Per-process maximum core file limit in units of 512 words for batch (`[b]`) or interactive (`[i]`) processes. This represents the maximum size of a core file that the process can create. If the size of the process is larger than this limit, a partial core file will be created. A partial core file contains just the user and user common structures. If *n* is `unlimited`, no restriction on the size of the core file is applied. If this field is not specified, the appropriate value from the default table is used. The release default is `unlimited`.

`pcpulim[b] :`*n*`:`
`pcpulim[i] :`*n*`:` Per-process CPU limit, in seconds, for batch (`[b]`) or interactive (`[i]`) processes. If *n* is `unlimited`, no restriction on the amount of CPU usage is applied. If this field is not specified, the appropriate value from the default table is used. The release default is `unlimited`.

`permbits =|+|- :`*octal*`:`
`permbits =|+|- :`*name*, *name*, ...`:`
User permission bits supported by Cray Research. The following names are recognized values, the octal equivalents, and the corresponding `libudb`(3C) bit names (for a description of valid `permbits`, see `libudb`(3C)):

Value	**Octal Value**	**`libudb`(3C) Bit Name**
`bypasslabel`	00000000001	`PERMBITS_BYPASSLABEL`
`realtime`	00000000002	`PERMBITS_REALTIME`
`nobatch`	00000000004	`PERMBITS_NOBATCH`
`noiactive`	00000000010	`PERMBITS_NOACTIVE`
`yp`	00000000020	`PERMBITS_YP`
CAUTION: If `yp` is on, password aging no longer functions (see `pwage` field).		
`plock`	00000000040	`PERMBITS_PLOCK`
`cpu-dedicate`	00000000100	`PERMBITS_DEDIC`
`acct`	00000000200	`PERMBITS_ACCT`
`suspend-resume`	00000000400	`PERMBITS_SUSPRES`
`resource`	00000001000	`PERMBITS_RESLIM`
`mount`	00000002000	`PERMBITS_MOUNT`
`system-param`	00000004000	`PERMBITS_SYSPARAM`
`chroot`	00000010000	`PERMBITS_CHROOT`
`sigany`	00000020000	`PERMBITS_SIGANY`

Value	Octal Value	libudb(3C) Bit Name
tape-manage	00000040000	PERMBITS_TAPEMANAGER
mknod	00000100000	PERMBITS_MKNOD
diag	00000200000	PERMBITS_DEVMAINT
nice	00000400000	PERMBITS_NICE
id-change	00001000000	PERMBITS_ID
acctid	00002000000	PERMBITS_ACCTID
system-restricted	00004000000	PERMBITS_RESTRICTED
chown	00010000000	PERMBITS_CHOWN
mlsmount	00020000000	Unused. This permbit is available to assign to user accounts, but it no longer grants special abilities.
guard	00040000000	PERMBITS_GUARD
wrunlab	00100000000	PERMBITS_WRUNLABEL
askacid	00200000000	PERMBITS_ASKACID
guest	00400000000	PERMBITS_GUEST
guestadm	01000000000	PERMBITS_GUESTADM
groupadm	02000000000	PERMBITS_GROUPADM
ipc-persist	04000000000	PERMBITS_IPCPERSIST

permits =|+|- :*name1*, *name2*, ..., *namen*:
UNICOS security permissions. Only an appropriately authorized user can change this field. For a description of valid permits, see libudb(3C).

pfdlimit[b] :*n*:
pfdlimit[i] :*n*: Per-process maximum open file limit for batch ([b]) or interactive ([i]) processes. This represents the maximum number of file descriptors that a process belonging to this user can allocate. At minimum, *n* must be at least the value of OPEN_MAX (64); udbsee(1) does not accept a value less than OPEN_MAX. In addition, the system imposes an upper limit of K_OPEN_MAX; udbgen allows values greater than K_OPEN_MAX, but the system forces the process' open file limit to K_OPEN_MAX. If this field is not specified, the appropriate value from the default table is used. The released default is 255.

pfilelim[b] :*n*:
pfilelim[i] :*n*: Per-process file allocation limit, in 512-word blocks, for batch ([b]) or interactive ([i]) processes. If *n* is 0 (default), there is no limit.

`pmemlim[b] :`*n*`:`
`pmemlim[i] :`*n*`:` Per-process memory limit, in 512-word blocks, for batch (`[b]`) or interactive (`[i]`) processes. If *n* is `unlimited`, no restriction on the amount of memory usage is applied. If this field is not specified, the appropriate value from the default table is used. The release default is `unlimited`.

`pmpptime[b] :`*n*`:`
`pmpptime[i] :`*n*`:` Per-process MPP maximum reservation time limit, in wall-clock seconds, for batch (`[b]`) or interactive (`[i]`) processes. (This field is present but not functional for Cray Research systems without a Cray MPP system.) If *n* is `unlimited`, no restriction on the MPP reservation time is applied. If *n* is `none`, the MPP system is inaccessible by the process. If this field is not specified, the appropriate value from the default table is used. The release default is `none`.

`psdslim[b] :`*n*`:`
`psdslim[i] :`*n*`:` Process secondary data segment limit in 512-word blocks for batch (`[b]`) or interactive (`[i]`) jobs. (This field is present but not functional for Cray Research systems without an SSD.) If *n* is `unlimited`, no restriction on the amount of secondary data segment usage is applied. If this field is not specified, the appropriate value from the default table is used. The release default is `unlimited`.

`pwage :force, superuser,` *max*`,` *min*`,` *time*`:`
`pwage :force, superuser,` *max*`,` *min*`,` +*age*`:`
`pwage :`*max*`,` *min*`:`

`pwage ::` Password age control fields are manipulated with `pwage`. The keywords `force` and `superuser` are used to set or clear the `PWFL_FORCE` and `PWFL_SUPERUSER` flags. If either keyword is preceded with a minus sign, the flag is turned off. If a keyword has a plus sign or no sign, the flag is turned on. For example, to set `PWFL_FORCE`, use the keyword `+force` or `force`. To turn off `PWFL_SUPERUSER`, use the keyword `-superuser`. If the `superuser` flag is turned on, password aging is to be enabled and only the super user is allowed to change the user's password. Both the force and super-user flags cannot be set on at the same time.

Flags that need not be changed should not be named with a keyword, because a keyword will force the on or off state; omitting the keyword will leave the flag as it is in the record. If a keyword is omitted, also omit its separating comma.

The *max*, *min*, and *time* fields control how old a password can become (*max*), how long it has to exist before being changed (*min*), and when it was changed (*time*). Neither *max* nor *min* may exceed 64 weeks. Either of these values may be expressed in units other than the default [`w`]eeks, namely [`d`]ays or [`s`]econds, although it must be possible to express the value as an integer. For example, a minimum of one week and a maximum of 12 weeks could be written: "`12, 1`", "`12w, 1w`", "`84d, 7d`", or "`7257600s, 604800s`". It is not necessary that both values have the same unit designator, although it is wise to use uniform units whenever possible, for readability. Internally the values are maintained in seconds. The value of *max* must never be less than the value of *min*. The *time* field is the value of the system time of day clock expressed as an integer.

The second form of `pwage` shown above is distinguished by a plus sign preceding the last numeric value. This causes age to be interpreted as the amount of time to subtract from the time "now" to result in a value of the time of day clock which is *age* units in the past and then store that value in the time field. Usually, this is intended to make it easy to set the current time in the field by using the value +0 as the age. Units are allowed on this value just as with *max* and *min*. For example, to set a time 14 days ago, write: "`,,+2`", "`,,+14d`", or "`,,+1209600s`". Note that two commas must precede the time if *max* and *min* ages are not specified, since this part of the directive is position-dependent, requiring all left context in order to determine the meaning of the particular value string.

The third form of `pwage` shown above is used to alter the *max* and *min* age fields.

The fourth form of `pwage` shown above is used only to remove age control from a record. All age control fields are set to a zero or null state, which removes age control totally. Once this has been done, all historical information is lost from the record.

Note that when the YP permbit is set (see the `permbits` field) and the password is being accessed from the database, password aging is disabled.

`resgrp :`*name*`:`
`resgrp :`*n*`:`
	The user name or user ID of the resource group to which this user belongs. That a user name has been specified is an initial assumption. If there is no existing record in the database with that name and the field is a number, the value is assumed to be a user ID. Care must be taken when using names during creation to ensure that any names used here have already been created in the database. The `udbsee`(1) command always provides user IDs to avoid this problem.

`root :`*directory*`:`	Login root directory; the root directory of the user's login process is set to *directory* with `chroot`(2). A string of up to 63 characters may be specified.
`shares :`*n*`:`	Allocated shares for the fair-share scheduler. The default for *n* is 0, which means that no shares are allocated.
`shcharge :`*vv.vv*`:`	Long-term accumulated costs for the fair-share scheduler. This is a floating-point value field. NOTE: The UDB floating-point value fields are stored in Cray Research format. If these fields are accessed from an IEEE CPU, the UDB library performs the necessary conversion.
`shell :`*sh_name*`:`	Default login shell. Up to 63 characters may be specified. The default value for *sh_name* is `/bin/sh`.
`shextime :`*n*`:`	Time last lnode was freed (for the fair-share scheduler).
`shflags =|+|- :`*octal*`:`	The fair-share scheduler `l_flags`.
`shusage :`*vv.vv*`:`	The fair-share scheduler decaying accumulated costs. This is a floating-point value field. NOTE: The UDB floating-point value fields are stored in Cray Research format. If these fields are accessed from an IEEE CPU, the UDB library performs the necessary conversion.

`sitebits =|+|- :`*octal*`:`
`sitebits =|+|- :`*name*, *name*, ...`:`
	User permission bits reserved by the site. Names `site1` (octal 01) through `site32` (octal 020000000000) are recognized values.

`trap :`*n*`:`	UNICOS security trap field. The default for *n* is 0, which means the user login is not trapped. If *n* is `1`, the user login is trapped. Only an appropriately authorized user can change this field.

`uid :`*n*`:`
`uid :next:` If a number is specified, that value is assigned to the user ID. If the value is `next`, the next higher user ID from the UDB is assigned to this user. The maximum value of a user ID is one less than the number returned from `sysconf(_SC_UID_MAX)` (a system call). This is presently 59999 (60000 – 1). Only an appropriately authorized user can change this field.

`valcat=|+|- :`*name1*`,` *name2*`,` ...`,` *namen*`:`
Authorized categories. Only an appropriately authorized user can change this field.

Global Default Table

The `global :default:` block introduction phrase indicates that the subsequent field names and field value pairs are to be assigned to the global default table. There are default table entries for the fields listed in the following table:

Field Name	**Released Default**
`jcpulim[b|i]`	`unlimited`
`pcpulim[b|i]`	`unlimited`
`jmemlim[b|i]`	`unlimited`
`pmemlim[b|i]`	`unlimited`
`jsdslim[b|i]`	`none`
`psdslim[b|i]`	`none`
`jfilelim[b|i]`	`unlimited`
`pfilelim[b|i]`	`unlimited`
`jtapelim[b|i][t]`	`none`
`nice[b|i]`	`0`
`jproclim[b|i]`	System's default
`jpelimit[b|i]`	`none`
`jmpptime[b|i]`	`none`
`jmppbarrier[b|i]`	NOTE: Not implemented. `none`
`pmpptime[b|i]`	`none`
`pcorelim[b|i]`	`unlimited`
`pfdlimit[b|i]`	`255`
`jshmsegs[b|i]`	`0`
`jshmsize[b|i]`	`0`

The release defaults are applied by `udbgen` when it updates a UDB that has a default table containing all zeroes. To create a default table in an existing UDB, execute `udbgen -c'#'`. This is an empty modification request, but it causes the default table to be created with the released defaults. To change one or more entry, write the appropriate directive line.

To set the interactive job CPU limit to 300 seconds and the interactive process memory limit to 8000 clicks, enter the following command:

```
global :default: jcpulim[i] :300: pmemlim[i] :8000:
```

The defaults from the table above are supplied when new records are created and the fields that have defaults are not named or have empty value fields (::). After the defaults have been applied to a record, there is no way to determine that the value has been derived from the default. This means that existing records will not track changes to the default values. To apply the defaults to an existing record, an update directive can be written as follows, where `user` is a known name in the UDB:

```
update :user: jcpulim[i] :: pmemlim[i] ::
```

Global Tape Name Map

The `global :tmap:` global tape name map provides a common place to store the names of the tape devices so the same names can be used everywhere. After names have been placed in the map, `udbgen` will accept the names in place of the ordinal (*t*) in the `jtapelim[b|i][`*t*`]` directives. Tape names are from 1 to 8 characters in length and are case-sensitive. Make sure these names match those known to the tape subsystem. Access to the map information is through the library routine `getudbtmap` (see `libudb`(3C)) and the interface is defined in `include/udb.h`. No names are defined by default, but the example below shows how to define the names `TAPE` and `CART` as ordinals `0` and `1`:

```
global :tmap: tmap[0] :TAPE: tmap[1] :CART:
```

Once this has been done, `jtapelim` directives can be written as follows:

```
update :user: jtapelim[b][CART] :2:
      jtapelim[b][TAPE] :1: jtapelim[i][CART] :1:
      jtapelim[i][TAPE] :none:
```

CAUTION: After the names have been created and are used, changing them can cause problems, because the names may become part of many users' files.

NOTES

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

Privilege Text	**Action**
`chgany`	Allowed to change all UDB fields.
`nosec`	Allowed to change all UDB fields, except sensitive fields.

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

Active Category	**Action**
`system`, `secadm`	Allowed to change all UDB fields.
`sysadm`	Allowed to change all UDB fields, except sensitive fields.

If the `PRIV_SU` configuration option is enabled, the super user is allowed to change all UDB fields.

MESSAGES

Many error messages are possible from `udbgen`. Generally, they fall into the following five categories:

- Warning and informative messages
- Usage errors
- Source errors
- I/O errors
- Messages from the UDB manager functions

The messages pinpoint the problem and should be self-evident. Errors found in the input file give the line number and, when possible, the user name and field in error. Source errors and warnings prohibit updating of the record in error on the database. All messages are sent to the `stderr` file. Usage messages exit with an error code of 1. If any warning messages occur, the exit code is 2 and fatal errors exit with a value of 3.

BUGS

The `-v` option does not catch all problems because the database is not actually updated.

It is possible to specify encrypted passwords that contain characters outside of the defined alphabet. This is allowed in order to handle old `/etc/passwd` and `/etc/uentry` files where deleted users remain in the file but with an encrypted password that is impossible to match.

When the special *uid* value `next` is used to assign the user ID, and multiple `udbgen` processes are active, the same user ID could be assigned to more than one user.

Because of the definition of the source language, a block is not fully processed in interactive mode until the next `create`, `delete`, `update`, or `quit` block is seen. If this happens, error messages for the previous block are issued after the first line of the next block is entered.

EXAMPLES

Example 1: Batch UDB creation. The following example is taken from the output of `udbsee`(1), which is designed to create an output file acceptable as input to `udbgen`. This example builds a new UDB if one does not exist or places the named definitions in an existing database, replacing any entries with a new definition.

```
#
#      Created by udbsee -a
#      Thu Apr 16 16:39:30 1992
#
create :fil:  uid    :64:
       comment       :F. I. Ling:
       passwd :TI..ekLKRkiZI:
       pwage  :-force, -superuser,
               12w, 1d, 703460104: #Max, min age, changed 04/16/92 16:35:04
       gids   :102,  64,    1,     2,     3,
               14,   15,    16:
       acids  :61,   16,    14,    13,    12,
               1:
       dir    :/w/fil:
#
create :jed:  uid    :66:
       comment       :John E. Doe:
       passwd :9VrQEF/E1MI.C:
       pwage  :+force, -superuser,
               4w, 3d, 703460104: #Max, min age, changed 04/16/92 16:35:04
       gids   :102,  66:
       acids  :317:
       dir    :/w/jed:
#
create :fff:  uid    :77:
       comment       :Frank F. Frost:
       passwd :jpAB9T5FZKnQA:
       pwage  :-force, +superuser,
               12w, 12w, 703460104: #Max, min age, changed 04/16/92 16:35:04
       gids   :101,  1027,  116:
       acids  :48:
       dir    :/u/fff:
```

Example 2: Batch UDB update. To change the maximum password age to four weeks counting from the present, write the source file shown below. The minimum value for user `fff` must be altered to satisfy the rule that maximum age must not be less than minimum age. For the other two users, minimum age is not changed.

```
update :fil: pwage:4w,,+0:
update :jed: pwage:4w,,+0:
update :fff: pwage:4w,4w,+0:
```

Example 3: Interactive entry creation. To add a new user named `jjj` to the UDB, either use an input file, as in the preceding example, or do this interactively from the keyboard, as follows:

```
$udbgen
udbgen: 1>create:jjj: uid:109: comment:Jane J. Jones:
udbgen: 2>passwd:EfgTYUnnjuKP:
udbgen: 3>pwage:+force,4,1:
udbgen: 4>dir:/w/jjj: gids:64,1: acids:297:
udbgen: 5>quit
Added 1 record
$
```

Example 4: Report generation (`-a` option). Running `udbgen` with the `-a` option invokes the `shrtree`(8) evaluation command. Sample output from the `-a` option is shown in this example. For an explanation of the output, see `shrtree`(8).

```
gust.1-> udbgen -aR

DISPLAY OF SHARE TREE
---------------------
UDB path:      DEFAULT
Analyzed:      By UID
Format:        Groups only
Maxgroups:     4
Node:          ALL
Group Count:   17
Account Count: 0
User Count:    1402
Warnings:      10
Errors:        0

Warning Count: 4     (Nc) Group has no references
Warning Count: 2     (Zs) User has zero shares
Warning Count: 4     (Zs) Group has zero shares

  Type      Name    ID    Status  Description
--------- -------- ----- -------- ---------------------------------
**WARN*** Serv      8001       10 Nc: Group has no references
**WARN*** Unknown   8393     1000 Zs: Group has zero shares
**WARN*** unknown     12     1001 Zs: User has zero shares
**WARN*** Country   8359     1010 Nc: Group has no references
**WARN*** Country   8359     1010 Zs: Group has zero shares
**WARN*** Region    8385     1010 Nc: Group has no references
**WARN*** Region    8385     1010 Zs: Group has zero shares
**WARN*** TechOps   8390     1010 Nc: Group has no references
**WARN*** TechOps   8390     1010 Zs: Group has zero shares
**WARN*** Tstusr01 33205     1001 Zs: User has zero shares
```

FILES

`/etc/acid`	Account name/ID file
`/etc/group`	Group name/ID/membership file
`/etc/passwd`	Password file
`/etc/udb`	User database file
`/etc/udb.public`	Copy of `/etc/udb` with public read permission. Sensitive information (such as the encrypted password and security fields) has been removed.
`/etc/udb_2/udb.index`	Public extension index file
`/etc/udb_2/udb.priva`	Private field extension file
`/etc/udb_2/udb.pubva`	Public field extension file

SEE ALSO

`shrsync`(8), `shrtree`(8)

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

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

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

Cray Data Migration Facility (DMF) Administrator's Guide, Cray Research publication SG–2135

General UNICOS System Administration, Cray Research publication SG–2301

NAME

`udbpl` – Prints administrative information for designated users

SYNOPSIS

`/etc/udbpl -a` [`-g`] [`-p` *udb_path*]
`/etc/udbpl` [`-p` *udb_path*] [*logins*] [*uid*[`-`*uid*]. . .]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `udbpl` command displays administrative information for the specified list of login names from the user database (UDB). Only authorized users can obtain sensitive information from the private UDB, `/etc/udb`. All other users are restricted to the public UDB, `/etc/udb.public`.

The `udbpl` command accepts the following options:

`-a` Prints information on all currently active users. If the `-g` option is also specified, the printed information is restricted to real users.

`-g` This option must not appear without the `-a` option. It restricts the information to be printed to real users (in other words, it does not print groups).

`-p` *udb_path* Directs `udbpl` to use an alternate UDB having the path name *udb_path*.

logins
uid[`-`*uid*]. . . If the `-a` option is not specified, one or more *logins*, *uids*, or *uid* ranges may be specified. The logins and user IDs must be separated by white space. A user ID range cannot include white space. For example, the range 150 to 500 must be entered as 150-500. To determine whether a *login* or *uid* has been specified, `udbpl` attempts to access the UDB record by name. If that attempt is unsuccessful, it examines the string for a minus sign indicating an inclusive range. If `udbpl` finds no range indicator, it attempts to access a record by user ID. If it finds a range indicator, the bounds of the range are determined and all records in the range are accessed by user ID.

If no options are specified, `udbpl` prints permitted administrative information for the login name of the person who executed the command.

Information Format

The information printed by `udbpl` is divided into four lists, as follows:

List	**Description**
Shares	The number of shares the user has within the scheduling group, the nominal share of the resources, the current effective share of the resources (taking into account the user's recent usage of the resources), and the normalized usage of the resources (in the range 1 through 1000)

Privileges	The user’s scheduling group and access groups, and a set of flags denoting account privileges and status
General	The encrypted password and the time the password was last changed (valid only if the user is privileged to see nonpublic administrative information), the total time the account has used the computer, the time the account was last used, the groups of terminals the account may log in, and other information that is site-dependent
Strings	The login name, its initial shell and directory, and a description of the owner of the account

NOTES

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

Active Category	**Action**
`system`, `secadm`, `sysadm`	Allowed to obtain sensitive information from the private UDB.

If the `PRIV_SU` configuration option is enabled, the super user is allowed to obtain sensitive information from the private UDB.

For `udbpl` to display accurate information, `share` must be turned on.

FILES

`/etc/udb*`	The user database. The path name can be changed by using the `-p` option.

SEE ALSO

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

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

`lnode`(5), `passwd`(5), `share`(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

`udbrstrict` – Enables and disables user access

SYNOPSIS

`/etc/udbrstrict -r` [`-d` *udb_dir*] [`-m` *RIB*] [`-f` *file*]
`/etc/udbrstrict -r` [`-d` *udb_dir*] [`-m` *RIB*] [`-F` *file*]
`/etc/udbrstrict -r` [`-d` *udb_dir*] [`-m` *RIB*] [`-l` *users*]
`/etc/udbrstrict -r` [`-d` *udb_dir*] [`-m` *RIB*] [`-L` *users*]
`/etc/udbrstrict -u` [`-d` *udb_dir*] [`-m` *RIB*] [`-f` *file*]
`/etc/udbrstrict -u` [`-d` *udb_dir*] [`-m` *RIB*] [`-F` *file*]
`/etc/udbrstrict -u` [`-d` *udb_dir*] [`-m` *RIB*] [`-l` *users*]
`/etc/udbrstrict -u` [`-d` *udb_dir*] [`-m` *RIB*] [`-L` *users*]
`/etc/udbrstrict -p` [`-d` *udb_dir*] [`-m` *RIB*]
`/etc/udbrstrict -P` [`-d` *udb_dir*] [`-m` *RIB*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `udbrstrict` command allows a privileged user (such as an operator) to enable and disable user access to the system. This includes both interactive access and batch access or some combination of the two. This capability is used to control access to the system during dedicated time.

Root (user ID 0) accounts are never restricted by `udbrstrict`. The `udbrstrict` command accepts two types of options: mode options, which control access and restriction, and user specification options, which specify users to be restricted or enabled.

The `udbrstrict` command accepts the following mode options and user specification options.

Mode Options

`-r` Restricts access. When used with no other options, it restricts access for all accounts except `root`. When used with the `-f` or `-l` options, it disables access for specified users. When used with the `-F` or `-L` options, it restricts access to specified users.

`-u` Removes access restrictions. When used with no other options, it removes access restrictions for all accounts (`root` is not affected). When used with the `-f` or `-l` option, it removes access restrictions for specified users. When used with the `-F` or `-L` option, it removes access restrictions for all users except those specified.

`-p` Displays a list of all restricted users.

`-P` Displays a list of all unrestricted users.

`-d` *udb_dir*
: Sets the path name for access to the user database (UDB) files. The default name is `/etc`. This option allows private or test versions of the UDB to be maintained. The path name must end with a directory name. Use of this option has no effect on the file specified using the `-f` or `-F` option.

`-m` *RIB* Specifies the restriction/unrestriction mode that is to be performed. The available modes are:

`R`	Restrict/unrestrict all access to the system
`I`	Restrict/unrestrict interactive sessions
`B`	Restrict/unrestrict batch sessions

User Specification Options

`-f` *file* Specifies an input file to be used as input. The names in the input file identify those users on whom the restrict/unrestrict operation is to be performed. The specified file must be formatted with one name per line. Input files may have white space and comments; white space (spaces and tabs) is ignored. A comment is any string preceded by a # character; it is in effect until the end of the current line.

`-F` *file* Specifies an input file to be used as input. The names in the input file identify those users on whom the restrict/unrestrict operation is not to be performed. The specified file must be formatted with one name per line. Input files may have white space and comments; white space (spaces and tabs) is ignored. A comment is any string preceded by a # character; it is in effect until the end of the current line.

`-l` *users* Specifies a list of users to be used as input. The user names in the list identify those users on whom the restrict/unrestrict operation is to be performed. The names must be separated by spaces. If the `-l` option is used, it must be the last option on the command line, and it must be followed by a list of one or more users.

`-L` *users* Specifies a list of users to be used as input. The names in the list identify those users on whom the restrict/unrestrict operation is not to be performed. The names must be separated by spaces. If the `-L` option is used, it must be the last option on the command line, and it must be followed by a list of one or more users.

NOTES

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

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

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

WARNINGS

The `udbrstrict` utility also disables Network Queuing System (NQS) and `cron` jobs. If NQS is started while the `udbrstrict -r` (or `udbrstrict -r -m B`) option is set, all checkpointed and all queued NQS jobs of all access-disabled users will be deleted.

MESSAGES

The following message appears when a user is prevented from logging in because the failing routine `setlimits` from `librsc` has only one possible exit.

```
setlimits: login restricted
```

EXAMPLES

Example 1: The following example restricts the use of the system to `root`.

```
udbrstrict -r
```

Example 2: The following example removes restrictions for all accounts.

```
udbrstrict -u
```

Example 3: The following example restricts the use of the system to users `mary` and `bob`.

```
udbrstrict -r -L mary bob
```

Example 4: The following example removes restrictions for all users except `bob`.

```
udbrstrict -u -L bob
```

Example 5: The following example restricts all batch access to the system.

```
udbrstrict -r -m B
```

Example 6: The following example displays a list of all users who have their interactive sessions disabled.

```
udbrstrict -p -m I
```

SEE ALSO

`udbgen`(8), `udbpl`(8)

`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

`unitap` – Provides online testing of magnetic tapes

SYNOPSIS

`/etc/diag/unitap`

IMPLEMENTATION

All Model E based Cray Research systems

DESCRIPTION

The `unitap` command provides online testing of magnetic tapes, either under the control of the tape daemon (up mode) or without the tape daemon (down mode). `unitap` can test up to eight tape units at one time. It is a menu-driven test that can also be run from the command line or by using an input file that directs input into `unitap`.

NOTE: The `unitap` command is not supported on GigaRing based Cray Research systems. Instead of `unitap`, use `vtt`(8) on GigaRing based systems. For more information, see the `vtt`(8) man page.

The `unitap` testing options are as follows:

- Quick confidence tests. This option executes selected tape confidence tests.
- All confidence tests. This option executes all of the tape confidence tests.
- Two- through eight-path conflict tests. This option executes a selection of tape tests in parallel to exercise from two through eight tape paths. The tests verify that the paths can withstand conflict.
- Canned tests. This option executes a user-selected test (from the Canned Tests menu).
- Test loop. This option executes a user-defined test (defined with the Programming tool).

When running `unitap`, you must be prepared either to mount the tape when requested or to start the test with a scratch tape in the device.

The `unitap` environment can be modified by setting several environment variables. If the `UNITAP_TPMNT_OPTION` variable is defined, its contents are appended to the `tpmnt`(1) command, allowing the addition of site-specific `tpmnt` options. The `UNITAP_TRACE_FILENAME` variable specifies the default name of the trace file created by `unitap`. The name will still default to `unitap.trace`.

The `UNITAP_SPECIAL_NAME` and `UNITAP_SPECIAL_NAME_`*n* variables specify the actual name of the tape group when you use the `SPECIAL` option to set the tape group name. If these variables are not used, the tape group will actually be `SPECIAL`, just like for `TAPE`, `CART`, `SILO`, `QIC`, `TEST`, `EXP`, `3490`, and `ER90`. The value of `UNITAP_SPECIAL_NAME` is used for all paths where `UNITAP_SPECIAL_NAME_`*n* is not set. The value of *n* determines which path (1 through 8) uses the `UNITAP_SPECIAL_NAME_`*n* value as the actual name of the tape group.

Tape units can be up or down when they are tested with `unitap`. When you are planning to test in down mode, enter the following form of the `tpconfig`(8) command to down a device: `tpconfig` *dev* `DOWN`. To test in down mode, you must have diagnostic privileges granted by your system administrator in your user database (UDB) or be running as super user. To test in up mode, you must have tape read and write privileges granted by your system administrator. However, you can practice using `unitap` without these privileges by running the test in learn mode or error mode. These modes simulate the passing and failing of test execution without removing the tape device from normal system operations. Choose learn mode by entering `L` or error mode by entering `E`.

By default, `unitap` runs in up mode. Entering the `CH` option from the Variable menu toggles `unitap` to run in down mode. Entering `DOWN` will also toggle `unitap` to down mode. Entering `UP` toggles `unitap` to up mode.

Each `unitap` menu option has a help window. To display a help window for a particular option, enter `HELP` followed by the option (for example, `help DBM`). `unitap` also has a program notes feature, which you can access by entering `W` from any menu. The program notes explain how to use `unitap`.

Whenever `unitap` is run, all input and output are stored in a file called `unitap.trace`. This file is useful for looking at failure history.

Entering the `/ce/bin/unitap` command displays the `unitap` Main menu. From this menu, you can display all the other menus in the `unitap` menu system.

`MN` Displays the `unitap` Main menu. The options on this menu include the following:

- `D` Displays the Debug menu.
- `T` Displays the Test menu.
- `V` Displays the Variable menu.
- `Q` Runs a quick test on the currently displayed path.
- `A` Runs all tests on the currently displayed path.
- `DUMP` Dumps all error data to the screen and to a trace file.
- `L` Toggles learn mode.
- `W` Displays a brief description of how to use `unitap`.
- `HELP` *option*
  Displays help information for *option* (for example, `HELP DBM`).
- `G` Displays the Global Options menu. The options on this menu are valid from all `unitap` menus.
- `EXIT` Exits `unitap` and releases channels dedicated to online diagnostic testing. You can also enter `bye` or `quit` to exit `unitap`.

`C` Displays the Canned Tests menu in up or down mode. This menu contains the tests that run on a single channel. These are the same tests used by the `Q`, `A`, and `2` through `8` options on the Test menu. The Canned Tests menu includes the following options:

`AC` All basic commands test. Tests the rewind, write, write tape mark, forward block, backward block, forward tape mark, and backward tape mark commands. This option does not test the read command. Applies only to down mode.

`BS` Bus test. Writes and reads 8-bit test patterns to the tape. Applies to both up and down mode.

`DT` Data buffer tests. Writes and reads 64-bit test patterns to the tape. Applies to both up and down mode.

`ET` End-of-tape test. Writes 32,768-byte blocks up to the value of `ETCOUNT` (the default is 20000). When the end-of-tape is detected, the tape is rewound and the data is verified. For this test, `unitap` goes into very fast mode, which disables the recording of system call history and the display of tape actions as they are processed. However, to help you determine that progress is being made, every hundredth read and write command is displayed. For this test, a block is 32,768 bytes. Applies only to down mode.

`ETCOUNT` *n*
Specifies how many blocks are written to the tape for the `ET` test. The default is 20000. Applies only to down mode.

`MV` Multivolume test. Writes 32,768-byte blocks up to the value of `MVCOUNT` (the default is 20000). After completing the writes, the data is read back and verified. For this test, `unitap` goes into very fast mode, which disables the recording of system call history and the display of tape actions as they are processed. However, to help you determine that progress is being made, every hundredth read and write command is displayed. For this test, a block is 32,768 bytes. Applies only to up mode.

`MVCOUNT` *n*
Specifies how many blocks are written to the tape for the `MV` test. The default is 20000. Applies only to up mode.

`BC` Byte counter test. Writes and reads 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, and 4096 bytes to the tape. Applies to both up and down mode.

`NB` Next byte counter test. Writes and reads 1 sector plus 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, and 4096 bytes to the tape. Applies to both up and down mode.

`BP` Tape block positioning commands test. Writes patterns to the tape, issues tape block positioning commands, and then reads the patterns to verify that the positioning commands work. Applies only to down mode.

`LA` Ladder tests. Writes and reads 1, 2, 3, 4, 5, 6, 7, and 8 sectors to the tape. Applies to both up and down mode.

`LW` Long write test. Writes 32768-byte blocks up to the value of `LWCOUNT` (the default is 5000). After completing the writes, the data is read back and verified. For this test, `unitap` goes into very fast mode, which disables the recording of system call history and the display of tape actions as they are processed. However, to help you determine that progress is being made, every hundredth read and write command is displayed. For this test, a block is 32,768 bytes. Applies to both up and down mode.

`LWCOUNT` *n*
Specifies how many blocks are written to the tape for the `LW` test. The default is 5000. Applies to both up and down mode.

`WC` Write compatibility test. Rewinds and writes a pattern to the tape. Applies to both up and down mode.

`RC` Read compatibility test. Rewinds and reads a pattern from the tape. Applies to both up and down mode.

`TP` Tape mark positioning commands test. Writes blocks and tape marks, issues tape-mark level positioning commands, then reads and verifies the patterns. Applies only to down mode.

`D` Displays the Debug menu, which contains troubleshooting tools. The options on this menu include the following:

`B` Displays the Breakpoint tool, which allows you to set a breakpoint immediately preceding or following a system call in a test.

`CB` Displays the Command Buffer tool, which allows you to build a string of `unitap` commands and save them to an internal variable.

`CD` Displays the Compare Data tool, which allows you to display the read and write data buffers, and exclusive ORs (logical differences) for the write and read address comparisons.

`DS` *n* Displays the Device Status menu, which shows information returned by the `tpbmx`(8) command. *n* can be a value from 1 through 15.

`E` Selects error mode, which is a subset of learn mode. In error mode, all tests fail (unlike learn mode, in which all tests pass). Error mode is useful for learning how to read failure conditions. `Error mode` is displayed in the upper left-hand corner of the screen.

`H` Displays the System Call History tool, which allows you to display a history of the last 15 system calls that preceded the current event.

`L` Selects learn mode. Learn mode simulates a passing test execution without removing the tape device from normal system operations. `Learn mode` is displayed in the upper left-hand corner of the screen.

`LO` Displays the Hardware Layout menu, which is a diagram that illustrates the typical hardware path from a Cray Research mainframe to a tape unit.

`M`	Displays the Memory tool, which allows you to display the read and write data buffers and modify the write data buffer.
`PG`	Displays the Programming tool, which allows you to build a test loop with up to 100 steps and up to 8 channels performing read, write, rewind, positioning, and compare operations.
`S`	Displays the Packet Status tool, which allows you to display the status of the last packet sent for each channel at the time of the current event.
`SB` *n*	Displays the Sense Byte menu. *n* is a value from 0 through 63. (You can also enter `SB` without a sense byte value to display a condensed version of sense byte information or enter `SBS` to display a summary of sense bytes.) This menu displays the meanings of additional bytes of information about a specific tape device.
`U`	Displays the User Tool menu, which enables you to issue various UNICOS commands without exiting `unitap`.

`G` Displays the Global Options menu. This menu displays all the options available in `unitap` in alphabetical order. These options are valid from all `unitap` menus. Entering `G` goes to the next page of the menu. At the end of the menu screens, you are returned to the first screen. The following options are unique to the Global Options menu:

`MAGIC`	Causes `unitap` to display the name of each internal routine as it is entered
`T2` *x*	Specifies that `unitap` should create a second trace file named *x*
`TIMESTAMP`	Prefixes most `unitap` output with the correct date and time
`TPBMX`	Displays operator information about tape devices
`TPCLR`	Clears the tape drive
`TPCONFIG`	Configures tape devices up and down
`TPFRLS`	Forcibly releases tape reservation and associated devices
`TPGSTAT`	Displays user reservation status for all users
`TPLABEL`	Labels magnetic tape reel or cartridge
`TPMLS`	Displays loader status
`TPMQL`	Displays the tape daemon’s mount request list
`TPU`	Unloads tape drives

`T` Displays the Test menu. The options on this menu include the following:

`Q`	Quick confidence tests. Runs selected tape tests.
`A`	All confidence tests. Runs all of the confidence tape tests.
`2 – 8`	Two- through eight-path confidence tests. Runs a selection of tape tests in parallel to exercise two through eight tape paths.

`C`	Canned tests. Executes a user-selected test.

`V` Displays the Variable menu in up or down mode. The options on this menu include the following:

`CH` *n*	Sets the channel number (20-37 octal). This option is required. Applies only to down mode. By default, `unitap` runs in up mode. Entering the `CH` option automatically toggles `unitap` to run in down mode.
`CO` *n*	Sets the controller number (O-F hexadecimal). Applies only to down mode. This option is required.
`CART`	Selects the 3480 cartridge tape group for up mode testing on the displayed path. By default, `unitap` selects the CART tape group. Applies only to up mode.
`3490`	Selects the 3490 cartridge tape group for up mode testing on the displayed path. Applies only to up mode.
`TAPE`	Selects the round-tape tape group for up mode testing on the displayed path. Applies only to up mode.
`SILO`	Selects the SILO tape group for up mode testing on the displayed path. Applies only to up mode.
`TEST`	Selects the TEST tape group for up mode testing on the displayed path. Applies only to up mode.
`QIC`	Selects the quarter-inch cartridge tape group for up mode testing on the displayed path. Applies only to up mode.
`ER90`	Selects the ER90 IPI tape group for up mode testing on the displayed path. Applies only to up mode.
`SPECIAL`	Selects the SPECIAL tape group for up mode testing on the displayed path. This option supports the use of a site-specific tape group name. Applies only to up mode.
`EXP`	Selects the EXP tape group for up mode testing on the displayed path. Applies only to up mode.
`DV` *n*	Specifies the name of the tape device to be tested (required for down mode; optional for up mode). *n* can be either the absolute path name (for example, `/dev/tape/300`) or just 300. When 300 is used, `unitap` will prefix it with `/dev/tape` when needed. Applies to both up and down mode.
`P`*n*	Initializes the path under test (channel, controller, and device). *n* is a value in the range 1 through 8. The default for *n* is 1. Applies to both up and down mode.
`PC` *n*	Sets the pass count (decimal). The default for *n* is 1. Applies to both up and down mode.
`RL`	Releases the dedicated path for the tape unit. Applies to both up and down mode.

`VSN` *x*	Changes the volume serial number on which to test. *x* can be up to a 6-digit volume serial number. Applies to both up and down mode.
`LB` *x*	Specifies the type of label on the tape. *x* is the 2-digit tape label type. Applies only to up mode.
`DOWN`	Toggles up and down mode.
`UP`	Toggles up and down mode.
`NW`	Toggles no write ring installed. Applies only to up mode.
`MD` *x*	Determines how `unitap` writes data to the tape. Applies only to up mode. *x* can be one of the following:

`TU`	Transparent unbuffered
`TB`	Transparent buffered

`IDRC`	Toggles IDRC (data compression). Applies to both up and down mode.
`F`	Toggles fast mode, which runs `unitap` faster than normal. This selection disables the Breakpoint tool and the System Call History tool. Fast mode status is displayed in the upper left-hand corner of the screen. Applies to both up and down mode.
`VF`	Toggles very fast mode, which runs `unitap` at maximum speed. This selection disables the Breakpoint tool and the System Call History tool. It also disables the echoing of `unitap` commands to the screen; no history trail will be available, and no activity will occur on the screen until the test completes or an error occurs. Applies to both up and down mode.
`R`	Returns to the previous menu.

`VSN1` through `VSN20`
Determines the list of VSNs on the `tpmnt`(1) command. (This menu option is not displayed on the Variable menu.) This option applies only to up mode.

`VOL1` through `VOL20`
Determines the list of VSNs on the `tpmnt`(1) command. (This menu option is not displayed on the Variable menu.) This option applies only to up mode.

NOTES

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

EXAMPLES

Example 1: Starts `unitap` in down mode; tests the path to the device from channel 20, controller 3, to device `tape100`; and runs a quick test (`q`) on device `tape100`. (With the exception of the *n* entry in the `DV` *n* option, `unitap` is not case-sensitive. It will accept entries in either uppercase or lowercase.)

```
/ce/bin/unitap ch 20 co 3 dv /dev/tape/tape100 q
```

Example 2: Starts `unitap` in up mode (the default) and runs a quick test (`q`) on the CART tape group (the default).

```
/ce/bin/unitap q
```

Example 3: Runs the two-channel conflict test in devices `00` and `01` and then exits the program. (For formatting purposes, the command in this example is shown on two lines; it should be entered as a one command.)

```
/ce/bin/unitap ch 20 co 0 dv /dev/tape/00 p2 ch 21 co 1
  dv /dev/tape/01 2 exit
```

SEE ALSO

`vtt`(8) for information on the online tape exerciser for GigaRing based Cray Research systems

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

NAME

`urmd` – Starts the Unified Resource Manager (URM) daemon

SYNOPSIS

`urmd` [`-D`] [`-l` *directory*] [`-p` *directory*] [`-s` *socket*] [`-t`] [`-u` *directory*] [`-c` *directive*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The following options and arguments to the `urmd` command are used for testing purposes only:

`-D`	Debug mode. Sends error messages to the `stderr` file.
`-l` *directory*	Alternate directory for the URM log file. By default, *directory* is `/usr/adm/urm`. This option overrides a directory specified with the `-p` option. When URM is running, use the `rmgr`(1) subcommand `set log directory` to change the location of the log file.
`-p` *directory*	Directory in which to run. This option is especially useful during development, so the daemon can run without interfering with the UNICOS operating system. Generally, use `-p` after moving to a working directory. Using the `-l` option overrides this directory.
`-s` *socket*	Socket name or port number to open as the service name for URM. The socket name will be passed as a `-s` *socket* option to the *command* named with the `-c` option. The default service name is `urm`.
`-t`	Test mode. This prevents `urmd` from creating new processes and no initiation command is run. (Even if `-c` is present, it is ignored). Use this only if you must use `cdbx` and want to prevent `urmd` from becoming a true daemon.
`-u` *directory*	UDB directory. This is usually not necessary because `urmd` only accesses the user database (UDB) in public-read mode. The reason it is present is to allow debugging such parts of `urmd` as the share evaluator without having to depend on the running UDB.
`-c` *directive*	The name of the directive used to initialize `urmd`. Defaults to `rmgr`. No command is run if either the `-t` option is used or the directive is `no`. Anything entered after the `-c` option is passed on as an argument to the `-c` *directive*.

NOTES

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

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

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

FILES

`/etc/config/daemons`	File to invoke `urmd` at system startup.
`/etc/config/urm/`	Directory containing configuration defaults for `urmd`. Do not attempt to alter any files in the directory, except the `configuration` file.
`/etc/config/urm/configuration`	File updated by the UNICOS installation and configuration menu system. Any site-specific URM configuration changes should be added at the end of this file.
`/etc/config/urm/init`	File containing the default `urmd` startup configuration directives. On startup of the `urmd`, `rmgr`(1) is automatically executed to include this file.
`/usr/adm/urm/Urm.`*yymmdd*	URM log file containing a record of URM activites for the day *yymmdd*.

SEE ALSO

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

UNICOS Resource Administration, Cray Research publication SG–2302

NAME

`urmsnap` – Captures the current URM configuration

SYNOPSIS

`/etc/urmsnap` [`-M` *rmgr_cmd*] [`-S` *socket*]

`/etc/urmsnap` [*input_file*]

IMPLEMENTATION

Cray PVP systems

DESCRIPTION

The `urmsnap` command captures the current Unified Resource Manager (URM) configuration. It reads the URM configuration, converts the information to `rmgr`(1) input format, and writes the result to standard output.

Changes to URM made on a running system with `rmgr`(1) are not permanent. `urmsnap` can be used to capture these changes for testing or for transfer to the permanent configuration of URM.

Options allow you to select a different version of the `rmgr`(1) command, a different socket, or an input file of `rmgr`(1) information (the output of the command `rmgr -c 'View /*'`). These options are useful only for internal testing purposes; they are not needed in normal use.

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

`-M` *rmgr_cmd*	Specifies an alternate pathname for the `rmgr`(1) command. This option is provided for internal testing; it is not recommended for general use. If the `-M` option is not used, the standard `rmgr`(1) command (`/bin/rmgr`) is used.
`-S` *socket*	Specifies the name of the socket connecting to URM. This option is provided for internal testing; it is not recommended for general use. If `-S` is not specified, the default URM socket (`urm`) is used.
input_file	Specifies an input file containing output from `rmgr`(1). This option is provided for internal testing; it is not recommended for general use. Use this option to save temporary URM configuration information for later examination. The content of this file must be identical to the output of the `rmgr`(1) `View` command (`rmgr -c 'View /*'`). This option cannot be used with the `-M` or `-S` option.

EXIT STATUS

If no error occurs, the `urmsnap` command exits with a return value of 0. Errors detected by `urmsnap`, such as option errors, result in an error message and an exit code of 1. Errors detected by `rmgr`(1) display the message text on standard error and result in a `urmsnap` exit code of 3. The exit code for `rmgr`(1) is reported in the `urmsnap` error message that follows the `rmgr`(1) message text.

SEE ALSO

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

UNICOS Resource Administration, Cray Research publication SG–2302

NAME

`usetjob` – Changes minimum rank field in URM job table entries

SYNOPSIS

`usetjob -r` *minrank* [`-D`] [`-M` *rmgr_command*] [`-p` *urmpath*] [`-S` *socket*] [`-W` *seconds*] *nqsjsn*[`.`*machine*] ...

IMPLEMENTATION

Cray PVP systems

DESCRIPTION

The `usetjob` command changes the minimum rank field in the Unified Resource Manager (URM) job table. You must be a URM administrator to alter this field. `usetjob` is an interface to the `rmgr`(1) command, which can also be used to accomplish this task.

By default, URM uses the priority determined by the Network Queuing System (NQS) for batch jobs. The minimum rank field of non-batch jobs is not used by URM; these jobs contain the default value of 0 in the minimum rank field.

The minimum rank, or NQS job priority, allows banding of batch jobs in the job backlog. Jobs are organized into categories that are independent of NQS queues. Jobs with higher minimum rank are guaranteed a better priority than jobs lower minimum rank. The `usetjob` command allows URM administrators to change the minimum rank outside of NQS.

A job is identified by the NQS sequence number, *nqsjsn*. Many IDs may be presented at once, each separated by white space. (The number of IDs is limited by the maximum length of the command line.) If jobs are being submitted from a number of machines, *nqsjsn* may not be unique. To further qualify the job name, you can specify either the machine name as reported by NQS or the numeric machine ID as shown by URM. The `usetjob` command translates machine names to machine IDs, using the translation file `/usr/adm/urm/midfile`, which is created by URM. This file is created the first time that NQS connects to URM, and it is updated on subsequent connections. If the translation file is not present, machine names are not accepted by `usetjob`.

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

`-r` *minrank*	Specifies minimum rank. The *minrank* argument must be a positive integer, 32 bits or less. This option is required.
`-D`	Specifies the debug option. Internal information is displayed on standard output (`stdout`). This option is useful only for internal testing purposes.
`-M` *rmgr_command*	Specifies the full path name of the `rmgr`(1) command. By default, the path name `/bin/rmgr` is used. This option is useful only for internal testing purposes.

`-p` *urmpath* Specifies the directory containing the machine-name-to-MID (machine identifier) translation file. (The translation file must be named `midfile`.) By default, the directory `/usr/adm/urm` is used. This option is useful only for internal testing purposes.

`-S` *socket* Specifies the socket name to pass to `rmgr`(1). By default, `rmgr`(1) uses the name specified by the environment variable `RMGR_SOCKET`, if set, or the default socket name `urm`. This option is useful only for internal testing purposes.

`-W` *seconds* Specifies the wait time in seconds for a child process to respond on a pipe. By default, the wait is 20 seconds. This option is useful only for internal testing purposes.

nqsjsn[`.`*machine*]
Specifies the job identification field, which is composed of a NQS job sequence number (JSN) with an optional machine name or machine identifier (MID). Only the JSN is required unless more than one machine is submitting jobs and the JSN values overlap; in this case, the machine name or MID is required. (The `usetjob` command issues a warning if overlapping JSNs are detected.)

Multiple job identification fields can be specified, separated by spaces. Jobs are processes in left-to-right order. An error for a job causes that field to be abandoned, and `usetjob` processes the next the next job identification field.

NOTES

Future releases of the UNICOS operating system will allow a URM administrator to change more fields in the URM job table.

ENVIRONMENT VARIABLES

The `RMGR_SOCKET` environment variable can be used to specify an alternate socket name for `rmgr`(1). By default, `rmgr`(1) uses the default socket name `urm`.

The `-S` option of `usetjob` overrides both `RMGR_SOCKET` and the default.

EXAMPLES

The following examples shows several basic uses of the `usetjob` command.

Example 1: This example changes the minimum rank for NQS job 12345.

```
usetjob -r 7 12345
```

Example 2: This example changes the minimum rank for NQS job 54321 from machine name `fred`. If multiple machines submit jobs, the machine name is necessary to prevent misinterpretation of the JSN.

```
usetjob -r 15 12345.fred
```

The machine ID (for example, 999) could be used instead of the machine name, giving a job identification field of `12345.999`.

FILES

`/usr/adm/urm/midfile`	Translation file; translates machine names to machine IDs

SEE ALSO

`urmd`(8)

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

UNICOS Resource Administration, Cray Research publication SG–2302

NAME

`vht` – Exercises SPN-based HIPPI devices (GigaRing based systems) or performs a test of a HIPPI channel (Model E systems)

SYNOPSIS

GigaRing based systems:

`/etc/diag/vht` [`-a` *action*] [`-b` *blocksize-max*[`:`*blocksize-min*]] | `pass`] [`-e` *echo-file-name*] [`-l` *logfile*] [`-o` *options*] [`-P` *pattern*] [`-p` *pass-end*[`:`*pass-start*[`:`*pass-step*]]] [`-T` *timeout*] [`-t` *tracefile*] `-I` *I-Field* `-u` *ULP-id* *device-name*

`/etc/diag/vht -h`

`/etc/diag/vht -V`

Model E systems:

`/etc/vht` [`-i` *idev*] [`-o` *odev*] [`-s` *spass*] [`-c` *epass*] [`-h` *hname*] [`-e` *filename*] [`-b` *blocksize*] [`-D`] [`-d`] [`-f`] [`-I` *I-Field*] [`-L`] [`-m` *errlim*] [`-n` *streams*] [`-P` *pattern*] [`-p`] [`-q`] [`-S`] [`-t` *timeout*] [`-x`] [`-y`] [`-l` *lpath*] [`-r`] [`-w`]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

GigaRing based systems:

The online exerciser for single-purpose node (SPN)-based HIPPI devices, `vht`, will exercise any SPN-based HIPPI network devices that have been configured into the operating system. Because `vht` uses raw channel HIPPI, the HIPPI interface must be configured "down" with `ifconfig`(8) if the device has been configured for use with TCP. (If you are not familiar with raw channel HIPPI, you are encouraged to use `vst`(8) to test the HIPPI device. `vst` requires that the TCP interface be configured "up" with `ifconfig`.)

Model E systems:

The `vht` command tests a HIPPI channel pair. The test creates a second process, by either a `fork`(2) operation or a remote shell command. The second process writes to the output device while the original one reads the input device. In each pass of the test, the I/O parameters (buffer lengths; synchronous, asynchronous, or `listio`; the number of buffers to make a block; and the data contents) are determined by pseudo-random numbers. The random number seed is the pass number, so that a given pass always executes the same test.

When running multiple instances of `vht`, the logical path option (`-l`) must be unique for each instance; either when in loopback mode (testing from the Cray host to the HIPPI switch) or when testing between two hosts. When testing between two hosts, their `-l` *lpath* options must match.

GigaRing Based Synopsis

The GigaRing based `vht` command accepts the following options:

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

`rw`	Read and write
`read`	Read
`write`	Write

The default is `rw`.

`-b` *blocksize-max*[`:`*blocksize-min*] | `pass`
Specifies the block size to be used for I/O operations. When the *blocksize-min* value is used, the actual block size will be randomly set to a value between *blocksize-min* and *blocksize-max* for each pass. When `pass` is specified, the block size will be set to the current pass count value.

`-e` *echo-file-name*
Writes the first data buffer to the specified file. No actual I/O to the HIPPI device is performed.

`-h`
Displays the command-line synopsis on `stderr`. `vht` then exits.

`-I` *I-Field*
Required option. Specifies the I-Field value. (If you are not familiar with the term "I-Field," you are encouraged to use `vst` instead.)

`-l` *logfile*
Specifies where the exerciser's log file should be written. The file specification can be either absolute or relative. By default, logging is disabled.

`-o` *options*
Specifies a colon-separated list of options to be turned on. Valid values are as follows:

`burst`	Deferred implementation. Uses the HIPPI short-burst-first feature for `listio`(2).
`debug`	Displays large amounts of information related to what the exerciser is doing.
`disconnect`	Causes HIPPI to disconnect after each packet.
`lock`	Locks the process(es) into memory to prevent swapping.
`fast`	Does not compare data.
`old`	Uses the old (pre GigaRing) `vht` command-line syntax.
`panic`	When a data miscompare occurs, panics the system.
`squeeze`	Deferred implementation.
`verbose`	Displays information about the progress of the exerciser.

By default, all options are turned off.

`-P` *pattern*
Specifies the data pattern to be used. Valid values are as follows:

`bits`	Each 64-bit word has a random sequence of consecutive 1-bits.
`slide0`	The first 64-bit word has all bits set except bit 0, which is cleared. Subsequent words are circularly left-shifted by one bit position.
`slide1`	The first 64-bit word has all bits cleared except bit 0, which is set. Subsequent words are circularly left-shifted by one bit position
`random`	Each 64-bit word is set to a random value.
`all`	All of the built-in data patterns are used, one per pass, in a circular fashion.

Instead of selecting one of the built-in patterns, you can specify a numeric constant that will be used to set each 64-bit word of the data buffer. The default is `all`.

`-p` *pass-end*[`:`*pass-start*[`:`*pass-step*]]
Specifies the number of passes to be performed. The number of passes must be the same on both the receiving and sending host. *pass-end* specifies the ending pass number; for example, `-p 1000` executes `vht` up to pass 1,000. *pass-start* specifies the starting pass number; for example, `-p 1000:500` executes `vht` starting at pass 500, and continuing up to pass 1,000. *pass-step* specifies how pass steps should be incremented; for example `-p 1000:500:100` executes `vht` starting at pass 500 and then steps up to pass 600, and so on, up to pass 1,000. *pass-start* and *pass-step* are most useful when specified with the `-b pass` option.

`-T` *timeout* Specifies the timeout value to be used when asynchronous I/O is performed.

`-t` *tracefile* Specifies where the exerciser's trace file should be written. The file specification can be either absolute or relative. By default, tracing is disabled.

`-u` *ULP-id* Required option. Specifies the ULP id value. (If you are not familiar with the term "ULP id," you are encouraged to use `vst` instead.)

`-V` Displays version information on `stderr`. `vht` then exits.

device-name Required option. Specifies the HIPPI device upon which the I/O operations are to be performed; for example, `/dev/hippi0/u0`. Note that only `/dev` is guaranteed to be the same on all machines.

Model E Synopsis

The Model E `vht` command accepts the following options:

`-i` *idev* Specifies the input path name of the channel to be tested. The default is `/dev/hippi0/i00`.

`-o` *odev* Specifies the output path name of the channel to be tested. The default is `/dev/hippi0/o00`.

NOTE: The path name to the channel pair to be tested can be changed by setting the environmental variable `VHTPREFIX`.

`-s` *spass* Specifies the starting pass number, *spass*.

`-c` *epass* Specifies the ending pass number, *epass*.

`-h` *hname* Specifies a remote host for the slave process that writes the output device. If the value is not specified, the local host is used, and the channels must be connected in loopback configuration.

`-e` *filename* Echoes data pattern to file (no test, does only one pass).

`-b` *blocksize* Overrides the normal random selection of packet size and forces all packets sent by `vht` to a given size. Specify *blocksize* in 64-bit words.

`-D` Disconnects after each packet (HIPPI).

`-d` Enables debug printouts; `vht` prints the parameters of each pass.

`-f` Sets fast-run mode; disables the exhaustive validation of received data.

`-I` *I-Field* Specifies the *I-Field* to use (HIPPI). The default *I-Field* is that given for the local host in the file `hycf.hippi`. The existence of the file is searched for in the following order: `./hycf.hippi`, `$HOME/hycf.hippi`, `/etc/hycf.hippi`, or, if no such file exists, then zero is used for the *I-Field*. Sample *I-Fields* are as follows: `-I 0x1300b00b` (hexadecimal) and `02300130013` (octal).

`-L` Locks the process in memory; that is, the `vht` test will not be swapped out of memory. You must have super-user privilege to use the `-L` option.

`-m` *errlim* Limits the number of data compare errors to print. The default is 10.

`-n` *streams* Specifies the number of streams. The default is 1. `vht` creates additional copies of itself by a `fork`(2) operation if multiple streams are requested. The input and output path names must end in a number, which `vht` increments each time it makes additional copies of itself, so each copy runs on a different logical path. When this option is specified, the output display may be garbled.

`-P` *pattern* Specifies the alternate data pattern to be used. Valid values are as follows:

`1`	Random bits (same as `-q`): each data bit has a 50% chance of being set or clear.
`2`	0x5555555555555555
`3`	0xAAAAAAAAAAAAAAAA
`4`	0x5555555555555555 0xAAAAAAAAAAAAAAAA
`5`	0x0000000000000000
`6`	0xFFFFFFFFFFFFFFFF
`7`	0x00000000FFFFFFFF
`8`	0xFFFFFFFFFFFFFFFF 0x0000000000000000
`9`	0xFFFFFFFFFFFFFFFF 0xFFFFFFFFFFFFFFFF 0x0000000000000000 0x0000000000000000

`10`	0x0001000100010001
`11`	0x0000010000000000 (position of 1-bit is random)
`12`	0x0000000000000001 0x0000000000000002 0x0000000000000003 etc.
`13`	0x55555555AAAAAAAA
`14`	0xAC89ACACAC89ACAC
`15`	0x870e1c3870e1c387 0x0e1c3870e1c3870e 0x1c3870e1c3870e1c 0x3870e1c3870e1c38 0x70e1c3870e1c3870 0xe1c3870e1c3870e1 0xc3870e1c3870e1c3
`16`	0x0043004300430043
`17`	0x0001020304050607 0x08090A0B0C0D0E0F 0x1011121314151617 etc.
`-1`	Select one of the above at random, different for each pass.
other	Default data pattern consisting of path number and 15 bits each of pass count, segment number, stride number and word number.

`-p` Panics the system on error. This option is ignored unless a `panic set` super-user command is active.

`-q` Randomizes test data. The default is to build test data words from the path number, pass count, segment number, stride number, and word number.

`-S` Uses SSD buffers. Can be used only if the system has an SSD.

`-t` *timeout* Specifies the time-out value in seconds for driver and HIPPI connection. Sixty seconds is the maximum.

`-x` Forces the data length to be equal to the pass number; pass 1 sends 1 word, pass 2 sends 2, and so on.

`-y` Tests sending a short burst first. By default, the HIPPI driver sends a short burst last.

The following options are used by `vht` when it calls itself by remote shell command, or when the companion process is started manually:

`-l` *lpath* Specifies a value, *lpath*, to be used in the *to* header field in output messages. Specify *lpath* as an 8-bit value. When running multiple instances of `vht`, `-l` must be unique for each instance; either when in loopback mode (testing from the Cray host to the HIPPI switch) or when testing between two hosts. When testing between two hosts, their `-l` *lpath* options must match.

`-r` Reads only the input device; does not spawn a companion process.

`-w` Writes only the output device; does not spawn a companion process.

NOTES

On any data error, `vht` writes two files in `/tmp`: one containing the expected data, and one containing the data actually received. These can be compared to help in failure analysis.

It is not recommended to use `vht` in multi-user mode unless the HIPPI interface is configured down or is in shared mode.

MESSAGES

Error messages are self-explanatory. For example, the following message appeared because the `-i` option specified a nonexistent device:

```
cray%  vht -i/dev/hippi/01
vht /dev/hippi/01:  No such file or directory
```

The next example failed because the `-r` option suppressed the second process, which would have written the output side, and the read timed out on pass 1. The error message lists the parameters selected for this pass.

The read was a `listio`(2) system call with two elements (segments) in the list, reading a single packet into two buffers of 370 and 1675 words. The write for pass 1 is a simple write of 17130 words, which is longer than the combined input buffers. The read process received timeout errors on both `listio` elements. It expected to recieve full byte counts on both buffers, with a nonfatal error on the second buffer indicating that excess data was discarded. `vht` dumped both the expected and recieved data to `/tmp/vht.3853o` and `/tmp/vht.3853i`, respectively.

```
cray% setenv VHTPREFIX /dev/hippi1
cray% vht -r
Selected logical device: /dev/hippi1/i00

------------------------------------------------------------
HIPPI test pass=1, path=128, pattern=0
Received - Input:  listio  370    1675 words
Sent - Output: write   17130 words
Path 128 pass 1 input segment 1:  I/O request timeout
/dev/hippi1/i00 HIPPI error code:  HIST_RTMO, Read request timeout
Path 128 pass 1 input segment 2:  I/O request timeout
/dev/hippi1/i00 HIPPI error code:  HIST_RTMO, Read request timeout
Input request unexpected status (nonfatal)
/dev/hippi1/i00 HIPPI error code:  HIST_RTMO, Read request timeout
(expected 01113, HIST_LONG, Long packet excess discarded)
dumping data buffers to /tmp/vht.3853i and /tmp/vht.3853o
vht 128 failed on pass 1
```

EXAMPLES

GigaRing based systems:

Example 1: The following example executes a loopback test for 1 pass only. On GigaRing systems the *I-Field* and *ULP-id* are required.

```
vht -a rw  -I 0x1300b00b -u 128
```

Example 2: The next example is almost the same, but it executes passes 1 through 10,000 with random data.

```
vht -a rw  -I 0x1300b00b -u 128 -p 10000  -P random
```

Example 3: The following example executes 1000 passes between `crayhost1` and `crayhost2`. These channels are connected. This example bypasses the exhaustive validation of input data.

```
crayhost1$     vht -a read -p 1000 -o fast

crayhost2$     vht -a write -p 1000 -o fast
```

Model E systems:

Example 1: The following example executes a loopback test for 1 pass only. It reads `/dev/hippi0/i01` and writes `/dev/hippi0/o01` using the default *I-Field* value.

```
cray%  vht -ii01 -oo01
```

Example 2: The next example is almost the same, but it executes passes 1 through 10,000 with random data.

```
cray%  vht -q -c 10000 -ii01 -oo01 -I 0x1300b00b
```

Example 3: The following example executes passes 100 through 1000 between `/dev/hippi0/i16` on `crayhost1` and `/dev/hippi0/o20` on `crayhost2`. These channels are connected. This example bypasses the exhaustive validation of input data.

```
crayhost1%  vht -s100 -c1000 -i/dev/hippi0/i16 -o/dev/hippi0/o20 -f -h crayhost2
```

FILES

GigaRing based systems:

Data miscompares are written to a file `/tmp/jtmp.*/*` in the following format:

```
A : expected
B : actual
x : logical difference

A (     0) + 7f 06 3c 38 7c fc 1f 06 06 0c 7f ff ff ff 1c 3e *..<8|..........>*
B (     0) + fe fd fb f7 ef df bf 7f fe fd fb f7 ef df bf 7f *................*
x (     0) + 81 fb c7 cf 93 23 a0 79 f8 f1 84 08 10 20 a3 41 *.....#.y.......A*
```

Model E systems:

`/dev/hippi*/*` received. These can be compared to help in failure analysis.

EXIT STATUS

The following exit codes apply to GigaRing based systems only:

Exit Code	Description
0	The command completed successfully with no I/O errors and no data miscompares.
1	The `close`(2) call failed. The error message will be displayed on `stderr`.
2	The `creat`(2) call failed. The error message will be displayed on `stderr`.
3	The `fcntl`(2) call failed. The error message will be displayed on `stderr`.
4	The `ioctl`(2) call failed. The error message will be displayed on `stderr`.
5	The `lseek`(2) call failed. The error message will be displayed on `stderr`.
6	The `open`(2) call failed. The error message will be displayed on `stderr`.
7	The `read`(2) call failed. The error message will be displayed on `stderr`.
8	The `reada`(2) call failed. The error message will be displayed on `stderr`.
9	The `stat`(2) call failed. The error message will be displayed on `stderr`.
10	The `write`(2) call failed. The error message will be displayed on `stderr`.
11	The `writea`(2) call failed. The error message will be displayed on `stderr`.
12	The `fclose`(3C) call failed. The error message will be displayed on `stderr`.
13	The `fflush`(3C) call failed. The error message will be displayed on `stderr`.
14	The `fgetpos`(3C) call failed. The error message will be displayed on `stderr`.
15	The `fopen`(3C) call failed. The error message will be displayed on `stderr`.
16	The `fprintf`(3C) call failed. The error message will be displayed on `stderr`.
17	The `fread`(3C) call failed. The error message will be displayed on `stderr`.
18	The `fscanf`(3C) call failed. The error message will be displayed on `stderr`.
19	The `fseek`(3C) call failed. The error message will be displayed on `stderr`.

20	The `fsetpos`(3C) call failed. The error message will be displayed on `stderr`.
21	The `ftell`(3C) call failed. The error message will be displayed on `stderr`.
22	The `ftruncate`(3C) call failed. The error message will be displayed on `stderr`.
23	The `fwrite`(3C) call failed. The error message will be displayed on `stderr`.
24	The `ffbksp`(3C) call failed. The error message will be displayed on `stderr`.
25	The `ffclose`(3C) call failed. The error message will be displayed on `stderr`.
26	The `fffcntl`(3C) call failed. The error message will be displayed on `stderr`.
27	The `fflistio`(3C) call failed. The error message will be displayed on `stderr`.
28	The `ffopen`(3C) call failed. The error message will be displayed on `stderr`.
29	The `ffpos`(3C) call failed. The error message will be displayed on `stderr`.
30	The `ffread`(3C) call failed. The error message will be displayed on `stderr`.
31	The `ffreada`(3C) call failed. The error message will be displayed on `stderr`.
32	The `ffseek`(3C) call failed. The error message will be displayed on `stderr`.
33	The `ffsetsp`(3C) call failed. The error message will be displayed on `stderr`.
34	The `ffweod`(3C) call failed. The error message will be displayed on `stderr`.
35	The `ffweof`(3C) call failed. The error message will be displayed on `stderr`.
36	The `ffwrite`(3C) call failed. The error message will be displayed on `stderr`.
37	The `ffwritea`(3C) call failed. The error message will be displayed on `stderr`.
38	The `accept`(2) call failed. The error message will be displayed on `stderr`.
39	The `bind`(2) call failed. The error message will be displayed on `stderr`.
40	The `connect`(2) call failed. The error message will be displayed on `stderr`.
41	The `gethostbyname`(3C) or `hostGetByName`() call failed. The error message will be displayed on `stderr`.
42	The `gethostname`(2) call failed. The error message will be displayed on `stderr`.
43	The `getsockopt`(2) call failed. The error message will be displayed on `stderr`.
44	The `listen`(2) call failed. The error message will be displayed on `stderr`.
45	The `recv`(2) call failed. The error message will be displayed on `stderr`.
46	The `send`(2) call failed. The error message will be displayed on `stderr`.
47	The `setsockopt`(2) call failed. The error message will be displayed on `stderr`.
48	The `shutdown`(2) call failed. The error message will be displayed on `stderr`.
49	The `socket`(2) call failed. The error message will be displayed on `stderr`.

`50`	The `fork`(2) call failed. The error message will be displayed on `stderr`.
`51`	The `kill`(2) call failed. The error message will be displayed on `stderr`.
`52`	The `signal`(2) call failed. The error message will be displayed on `stderr`.
`100`	The data read does not match the expected data. The differences are written to the log file.
`101`	A read request was made but not all data was received.
`102`	A write request was made but not all data was sent.
`255`	An error was encountered while parsing the command-line options. The command synopsis will be displayed on `stderr`.

SEE ALSO

GigaRing based systems:

`ifconfig`(8)

Concurrent Maintenance Tools User's Guide, publication SD–2627

Model E systems:

`panic`(8)

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

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

NAME

`vst` – Exercises network devices capable of running TCP/IP

SYNOPSIS

All Cray Research systems:

`/etc/diag/vst` [`-a` *action*] [`-b` *blocksize-max*[`:`*blocksize-min*] | `pass`] [`-e` *echo-file-name*] [`-H` *host-name*[`:`*port-number*]] [`-l` *logfile*] [`-n` *nprocs*] [`-o` *options*] [`-P` *pattern*] [`-p` *pass-end*[`:`*pass-start*][:*pass-step*]] [`-T` *timeout*] [`-t` *tracefile*]

`/etc/diag/vst -h`

`/etc/diag/vst -V`

Cray system workstation (SWS):

`/opt/CYRIccmt/bin/vst` [`-a` *action*] [`-b` *blocksize-max*[`:`*blocksize-min*] | `pass`] [`-e` *echo-file-name*] [`-H` *host-name*[`:`*port-number*]] [`-l` *logfile*] [`-n` *nprocs*] [`-o` *options*] [`-P` *pattern*] [`-p` *pass-end*[`:`*pass-start*][:*pass-step*]] [`-T` *timeout*] [`-t` *tracefile*]

`/opt/CYRIccmt/bin/vst -h`

`/opt/CYRIccmt/bin/vst -V`

IMPLEMENTATION

All Cray Research systems
Cray system workstation (SWS)

DESCRIPTION

The online, socket-based, device-independent network exerciser, `vst`, will exercise any network device that supports TCP/IP and has been configured into the operating system.

`vst` can be used to exercise TCP/IP network devices between two Cray systems, or between the System Workstation (SWS) and the Cray system (GigaRing only), or between the SWS and the multipurpose node (MPN) (GigaRing only). The general procedure is to open a listening connection on one platform (using the `-a read` option) and then attempt to send data to that connection (using the `-a write` option). With the exception of the MPN, it is possible to send and receive data on the same host.

Block size, patterns, and the number of passes must be the same on both the receiving and sending hosts.

The `vst` command accepts the following options:

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

`ping` Uses the Internet Control Message Protocol's (ICMP) `ECHO_REQUEST` packets to elicit `ECHO_RESPONSE` packets from the specified host, similar to the `ping`(8) command. The block size is limited to 16,376 bytes. SWS, UNICOS, or UNICOS/mk systems only (not valid on the MPN).

`rw` Reads and writes data on the same machine. SWS, UNICOS, or UNICOS/mk systems only (not valid on the MPN).

`read` Listens for and reads data from the cooperating process. This action must be started before the `vst -a write` action is started.

`write` Writes data to the host specified by the `-H` option (the default host is the machine on which `vst` is run). The cooperating `vst -a read` process must have been started first.

The default is `rw` on the SWS, UNICOS, and UNICOS/mk systems. On the MPN, the default is `read`.

`-b` *blocksize-max*[`:`*blocksize-min*] | `pass`
Specifies the block size to be used for I/O operations. When the *blocksize-min* value is used, the actual block size will be randomly set to a value between *blocksize-min* and *blocksize-max* for each pass. The maximum block size must be the same on both the receiving and sending host. When `pass` is specified, the block size will be set to the current pass count value. The block size is limited to 16376 bytes for the `ping` action.

`-e` *echo-file-name*
Specifies that the first data buffer is to be written to the specified file. No actual I/O to a socket is performed.

`-H` *host-name*[`:`*port-number*]
Specifies the name of the host on which the cooperating process should be located. The optional port number specifies the TCP/IP port to be used.

`-h` Displays the command-line synopsis on `stderr`. `vst` then exits.

`-l` *logfile*
Specifies where the `vst` log file should be written. The file specification can be either absolute or relative. By default, logging is disabled.

`-n` *nprocs*
Specifies how many `vst` processes are to be started. The default is one process. This option can be used to stress a device by increasing data throughput to or from the device. This option is not available in VxWorks.

`-o` *options*
Specifies a colon-separated list of options to be turned on. Valid values are as follows:

`debug`	Displays large amounts of information related to what `vst` is doing.
`fast`	Does not compare data.

`lock`	Deferred implementation. Locks the process(es) into memory to prevent swapping. UNICOS or UNICOS/mk systems only.
`panic`	When a data miscompare occurs, panics the system. UNICOS or UNICOS/mk systems only.
`squeeze`	Deferred implementation. UNICOS or UNICOS/mk systems only.
`ssd`	Deferred implementation. Places the data buffers in SSD (Cray PVP systems only). UNICOS or UNICOS/mk systems only.
`verbose`	Displays information about the progress of `vst`. This argument is valid only with the `rw` or `read` action. It has no effect on the `write` action.

By default, all options are turned off.

`-P` *pattern*

Specifies the data pattern to be used. Valid values are as follows:

`bits`	Each 64-bit word has a random sequence of consecutive 1-bits.
`slide0`	The first 64-bit word has all bits set except bit 0, which is cleared. Subsequent words are circularly left-shifted by one bit position.
`slide1`	The first 64-bit word has all bits cleared except bit 0, which is set. Subsequent words are circularly left-shifted by one bit position.
`random`	Each 64-bit word is set to a random value.
`all`	All of the built-in data patterns are used, one per pass, in a circular fashion.

Instead of selecting one of the built-in patterns, you can specify a numeric constant that will be used to set each 64-bit word of the data buffer. The default is `all`. Patterns must be the same on both the receiving and sending host.

`-p` *pass-end*[`:`*pass-start*[:*pass-step*]]

Specifies the number of passes to be performed. The number of passes must be the same on both the receiving and sending host. *pass-end* specifies the ending pass number; for example, `-p 1000` executes `vst` up to pass 1,000. *pass-start* specifies the starting pass number; for example, `-p 1000:500` executes `vst` starting at pass 500, and continuing up to pass 1,000. *pass-step* specifies how pass steps should be incremented; for example, `-p 1000:500:100` executes `vst` starting at pass 500, and then steps up to pass 600, and so on, up to pass 1,000. *pass-start* and *pass-step* are most useful when specified with the `-b pass` option.

`-T` *timeout*

Specifies the time-out value to be used when asynchronous I/O is performed. The default is 10 seconds.

`-t` *tracefile*

Specifies where the `vst` trace file should be written. The file specification can be either absolute or relative. By default, tracing is disabled.

`-V` Displays version information on `stderr`. `vst` then exits.

NOTES

For the SWS, the path to this command and the appropriate environment variable settings are provided by the `craydiag` and `crayadm` modules.

EXIT STATUS

Exit Code	**Description**
`0`	The command completed successfully with no I/O errors and no data miscompares.
`1`	The `close`(2) call failed. The associated error message text will be displayed on `stderr`.
`2`	The `creat`(2) call failed. The associated error message text will be displayed on `stderr`.
`3`	The `fcntl`(2) call failed. The associated error message text will be displayed on `stderr`.
`4`	The `ioctl`(2) call failed. The associated error message text will be displayed on `stderr`.
`5`	The `lseek`(2) call failed. The associated error message text will be displayed on `stderr`.
`6`	The `open`(2) call failed. The associated error message text will be displayed on `stderr`.
`7`	The `read`(2) call failed. The associated error message text will be displayed on `stderr`.
`8`	The `reada`(2) call failed. The associated error message text will be displayed on `stderr`.
`9`	The `stat`(2) call failed. The associated error message text will be displayed on `stderr`.
`10`	The `write`(2) call failed. The associated error message text will be displayed on `stderr`.
`11`	The `writea`(2) call failed. The associated error message text will be displayed on `stderr`.
`12`	The `fclose`(3C) call failed. The associated error message text will be displayed on `stderr`.
`13`	The `fflush`(3C) call failed. The associated error message text will be displayed on `stderr`.
`14`	The `fgetpos`(3C) call failed. The associated error message text will be displayed on `stderr`.
`15`	The `fopen`(3C) call failed. The associated error message text will be displayed on `stderr`.
`16`	The `fprintf`(3C) call failed. The associated error message text will be displayed on `stderr`.
`17`	The `fread`(3C) call failed. The associated error message text will be displayed on `stderr`.
`18`	The `fscanf`(3C) call failed. The associated error message text will be displayed on `stderr`.
`19`	The `fseek`(3C) call failed. The associated error message text will be displayed on `stderr`.

20	The `fsetpos`(3C) call failed. The associated error message text will be displayed on `stderr`.
21	The `ftell`(3C) call failed. The associated error message text will be displayed on `stderr`.
22	The `ftruncate`(3C) call failed. The associated error message text will be displayed on `stderr`.
23	The `fwrite`(3C) call failed. The associated error message text will be displayed on `stderr`.
24	The `ffbksp`(3C) call failed. The associated error message text will be displayed on `stderr`.
25	The `ffclose`(3C) call failed. The associated error message text will be displayed on `stderr`.
26	The `fffcntl`(3C) call failed. The associated error message text will be displayed on `stderr`.
27	The `fflistio`(3C) call failed. The associated error message text will be displayed on `stderr`.
28	The `ffopen`(3C) call failed. The associated error message text will be displayed on `stderr`.
29	The `ffpos`(3C) call failed. The associated error message text will be displayed on `stderr`.
30	The `ffread`(3C) call failed. The associated error message text will be displayed on `stderr`.
31	The `ffreada`(3C) call failed. The associated error message text will be displayed on `stderr`.
32	The `ffseek`(3C) call failed. The associated error message text will be displayed on `stderr`.
33	The `ffsetsp`(3C) call failed. The associated error message text will be displayed on `stderr`.
34	The `ffweod`(3C) call failed. The associated error message text will be displayed on `stderr`.
35	The `ffweof`(3C) call failed. The associated error message text will be displayed on `stderr`.
36	The `ffwrite`(3C) call failed. The associated error message text will be displayed on `stderr`.
37	The `ffwritea`(3C) call failed. The associated error message text will be displayed on `stderr`.
38	The `accept`(2) call failed. The associated error message text will be displayed on `stderr`.

`39`	The `bind`(2) call failed. The associated error message text will be displayed on `stderr`.
`40`	The `connect`(2) call failed. The associated error message text will be displayed on `stderr`.
`41`	The `gethostbyname`(3C) or `hostGetByName`() call failed. The associated error message text will be displayed on `stderr`.
`42`	The `gethostname`(2) call failed. The associated error message text will be displayed on `stderr`.
`43`	The `getsockopt`(2) call failed. The associated error message text will be displayed on `stderr`.
`44`	The `listen`(2) call failed. The associated error message text will be displayed on `stderr`.
`45`	The `recv`(2) call failed. The associated error message text will be displayed on `stderr`.
`46`	The `send`(2) call failed. The associated error message text will be displayed on `stderr`.
`47`	The `setsockopt`(2) call failed. The associated error message text will be displayed on `stderr`.
`48`	The `shutdown`(2) call failed. The associated error message text will be displayed on `stderr`.
`49`	The `socket`(2) call failed. The associated error message text will be displayed on `stderr`.
`50`	The `fork`(2) call failed. The associated error message text will be displayed on `stderr`.
`51`	The `kill`(2) call failed. The associated error message text will be displayed on `stderr`.
`52`	The `signal`(2) call failed. The associated error message text will be displayed on `stderr`.
`100`	The data read does not match the expected data. The differences are written to the log file.
`101`	A read request was made but not all data was received.
`102`	A write request was made but not all data was sent.
`255`	An error was encountered while parsing the command-line options. The command synopsis will be displayed on `stderr`.

SEE ALSO

SWS-ION Administration and Operations Guide, publication SG–2204

NAME

`vtt` – Provides tape device testing

SYNOPSIS

`/etc/diag/vtt` [`-a` *action*] [`-b` *blocksize-max*[`:`*blocksize-min*] | `pass`] [`-e` *echo-filename*] [`-l` *logfile*] [`-o` *options*] [`-P` *pattern*] [`-p` *pass-end*[`:`*pass-start*[`:`*pass-step*]]] [`-T` *timeout*] [`-t` *tracefile*] *device-name*

`/etc/diag/vtt -h`

`/etc/diag/vtt -V`

IMPLEMENTATION

GigaRing based Cray Research systems

DESCRIPTION

The online tape exerciser, `vtt`, will exercise any online tape devices that have been configured into the operating system. The devices may be configured either down or up. When using devices that are configured down, the block-special node in `/dev/tape` will be accessed directly. For devices that are configured up, device access will go through the tape daemon. For up mode tests, the *device-name* is assumed to be configured up via `rsv`(1) and `tpmnt`(1).

The `vtt` command accepts the following options:

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

`rw` Write a number of blocks, rewind the tape, read back the same number of blocks, and compare them to expected data values.

`read` Read a number of blocks and compare them to expected data values.

`write` Write a number of blocks.

`thrash`
Write a single block, rewind the tape, read back the block, and compare to expected data values. This sequence is repeated for each pass.

`eot` Write data until the end-of-tape is reached, rewind the tape, read back all the blocks, and compare them to expected data values. This test is only available if operating on a tape device that has been configured down (that is, `/dev/tape/*`).

`mv` Write data until a volume switch is performed, rewind the tape, read back all the blocks, and compare them to expected data values. This test is only available if operating on a tape device that has been configured up (that is, obtained via `rsv`(1) and `tpmnt`(1)).

`block`	Use block positioning operations to skip around the tape to write, read, and compare data values. This test is only available if operating on a tape device that has been configured up (that is, obtained via `rsv`(1) and `tpmnt`(1)).
`tapemark`	Use tapemark positioning operations to skip around the tape to write, read, and compare data values. This test is only available if operating on a tape device that has been configured down (that is, `/dev/tape/*`).

The default is `rw`.

`-b` *blocksize-max*[`:`*blocksize-min*] | `pass`
Specifies the block size to be used for I/O operations. When the *blocksize-min* value is used, the actual block size will be randomly set to a value between *blocksize-min* and *blocksize-max* for each pass. When `pass` is specified, the block size will be set to the current pass count value.

`-e` *echo-file-name*
Specifies that the first data buffer is to be written to the file. No actual I/O to the device is performed.

`-h` Displays the command-line synopsis on `stderr`. `vtt` then exits.

`-l` *logfile*
Specifies where the exerciser's log file should be written. The file specification can be either absolute or relative. By default, logging is disabled.

`-o` *options*
Specifies a colon-separated list of options to be turned on. Valid values are as follows:

`debug`	Displays large amounts of information related to what the exerciser is doing.
`fast`	Does not compare data.
`panic`	When a data miscompare occurs, panics the system.
`verbose`	Displays information about the progress of the exerciser.

By default, all options are turned off.

`-P` *pattern*
Specifies the data pattern to be used. Valid values are as follows:

`bits`	Each 64-bit word has a random sequence of consecutive 1-bits.
`slide0`	The first 64-bit word has all bits set except bit 0, which is cleared. Subsequent words are circularly left-shifted by one bit position.
`slide1`	The first 64-bit word has all bits cleared except bit 0, which is set. Subsequent words are circularly left-shifted by one bit position.
`random`	Each 64-bit word is set to a random value.
`all`	All of the built-in data patterns are used, one per pass, in a circular fashion.

Instead of selecting one of the built-in patterns, the user can specify a numeric constant that will be used to set each 64-bit word of the data buffer. The default is `all`.

`-p` *pass-end*[`:`*pass-start*[`:`*pass-step*]]
Specifies the number of passes to be performed (`thrash`) or blocks to be written/read (`rw`, `read`, `write`). This option is ignored for all other actions. The specification used when reading a tape should be the same as the specification used when writing the tape. *pass-end* specifies the ending pass number; for example, `-p 1000` executes `vtt` up to pass 1,000. *pass-start* specifies the starting pass number; for example, `-p 1000:500` executes `vtt` starting at pass 500, and continuing up to pass 1,000. *pass-step* specifies how pass steps should be incremented; for example `-p 1000:500:100` executes `vtt` starting at pass 500 and then steps up to pass 600, and so on, up to pass 1,000. *pass-start* and *pass-step* are most useful when specified with the `-b pass` option.

`-T` *timeout*
Specifies the timeout value to be used when asynchronous I/O is performed.

`-t` *tracefile*
Specifies where the exerciser's trace file should be written. The file specification can be either absolute or relative. By default, tracing is disabled.

`-V` Displays version information on `stderr`. `vtt` then exits.

device-name
Required option. Specifies the tape device upon which the I/O operations are to be performed.

EXIT STATUS

Exit Code	**Description**
0	The command completed successfully with no I/O errors and no data miscompares.
1	The `close`(2) call failed. The associated error message text will be displayed on `stderr`.
2	The `creat`(2) call failed. The associated error message text will be displayed on `stderr`.
3	The `fcntl`(2) call failed. The associated error message text will be displayed on `stderr`.
4	The `ioctl`(2) call failed. The associated error message text will be displayed on `stderr`.
5	The `lseek`(2) call failed. The associated error message text will be displayed on `stderr`.
6	The `open`(2) call failed. The associated error message text will be displayed on `stderr`.
7	The `read`(2) call failed. The associated error message text will be displayed on `stderr`.
8	The `reada`(2) call failed. The associated error message text will be displayed on `stderr`.
9	The `stat`(2) call failed. The associated error message text will be displayed on `stderr`.
10	The `write`(2) call failed. The associated error message text will be displayed on `stderr`.
11	The `writea`(2) call failed. The associated error message text will be displayed on `stderr`.

12	The `fclose`(3C) call failed. The associated error message text will be displayed on `stderr`.
13	The `fflush`(3C) call failed. The associated error message text will be displayed on `stderr`.
14	The `fgetpos`(3C) call failed. The associated error message text will be displayed on `stderr`.
15	The `fopen`(3C) call failed. The associated error message text will be displayed on `stderr`.
16	The `fprintf`(3C) call failed. The associated error message text will be displayed on `stderr`.
17	The `fread`(3C) call failed. The associated error message text will be displayed on `stderr`.
18	The `fscanf`(3C) call failed. The associated error message text will be displayed on `stderr`.
19	The `fseek`(3C) call failed. The associated error message text will be displayed on `stderr`.
20	The `fsetpos`(3C) call failed. The associated error message text will be displayed on `stderr`.
21	The `ftell`(3C) call failed. The associated error message text will be displayed on `stderr`.
22	The `ftruncate`(3C) call failed. The associated error message text will be displayed on `stderr`.
23	The `fwrite`(3C) call failed. The associated error message text will be displayed on `stderr`.
24	The `ffbksp`(3C) call failed. The associated error message text will be displayed on `stderr`.
25	The `ffclose`(3C) call failed. The associated error message text will be displayed on `stderr`.
26	The `fffcntl`(3C) call failed. The associated error message text will be displayed on `stderr`.
27	The `fflistio`(3C) call failed. The associated error message text will be displayed on `stderr`.
28	The `ffopen`(3C) call failed. The associated error message text will be displayed on `stderr`.
29	The `ffpos`(3C) call failed. The associated error message text will be displayed on `stderr`.
30	The `ffread`(3C) call failed. The associated error message text will be displayed on `stderr`.
31	The `ffreada`(3C) call failed. The associated error message text will be displayed on `stderr`.

32	The `ffseek`(3C) call failed. The associated error message text will be displayed on `stderr`.
33	The `ffsetsp`(3C) call failed. The associated error message text will be displayed on `stderr`.
34	The `ffweod`(3C) call failed. The associated error message text will be displayed on `stderr`.
35	The `ffweof`(3C) call failed. The associated error message text will be displayed on `stderr`.
36	The `ffwrite`(3C) call failed. The associated error message text will be displayed on `stderr`.
37	The `ffwritea`(3C) call failed. The associated error message text will be displayed on `stderr`.
38	The `accept`(2) call failed. The associated error message text will be displayed on `stderr`.
39	The `bind`(2) call failed. The associated error message text will be displayed on `stderr`.
40	The `connect`(2) call failed. The associated error message text will be displayed on `stderr`.
41	The `gethostbyname`(3C) or `hostGetByName`() call failed. The associated error message text will be displayed on `stderr`.
42	The `gethostname`(2) call failed. The associated error message text will be displayed on `stderr`.
43	The `getsockopt`(2) call failed. The associated error message text will be displayed on `stderr`.
44	The `listen`(2) call failed. The associated error message text will be displayed on `stderr`.
45	The `recv`(2) call failed. The associated error message text will be displayed on `stderr`.
46	The `send`(2) call failed. The associated error message text will be displayed on `stderr`.
47	The `setsockopt`(2) call failed. The associated error message text will be displayed on `stderr`.
48	The `shutdown`(2) call failed. The associated error message text will be displayed on `stderr`.
49	The `socket`(2) call failed. The associated error message text will be displayed on `stderr`.
50	The `fork`(2) call failed. The associated error message text will be displayed on `stderr`.
51	The `kill`(2) call failed. The associated error message text will be displayed on `stderr`.
52	The `signal`(2) call failed. The associated error message text will be displayed on `stderr`.
100	The data read does not match the expected data. The differences are written to the log file.

`101`	A read request was made but not all data was received.
`102`	A write request was made but not all data was sent.
`255`	An error was encountered while parsing the command-line options. The command synopsis will be displayed on `stderr`.

NAME

`wall` – Writes to all users

SYNOPSIS

`/etc/wall` [`-d` *secs*] [`-g` *group*] [*file*]

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The `wall` command reads standard input, or a specified *file*, until an end-of-file (EOF) is reached. It then sends the message written to standard input to all users currently logged-in, preceded by the following text:

`Broadcast Message from` *user* `at` *nodename date*

A typical use of the `wall` message is to warn all users before shutting down the system.

The `wall` command accepts the following options and arguments:

`-d` *secs*	The `-d` option specifies the number of seconds `wall` will wait for the message to be sent to all logged-in users. `wall` will exit after this amount of time even though the message was not delivered to all logged-in users. The default is 60 seconds.
`-g` *group*	The `-g` option allows a message to be sent only to members of the *group* specified. The *group* is defined in the `/etc/group` file (see `group`(5)).

The sender must have super-user status to override any protections the users may have invoked (see `mesg`(1)).

NOTES

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

Active category	**Action**
`system`, `secadm`, `sysadm`	Allowed to write to all users.

If the `PRIV_SU` configuration option is enabled, the super user is allowed to write to all users.

MESSAGES

`Cannot send to` *user*`...`

This message occurs when the `open`(2) system call on a user's tty file fails.

FILES

`/etc/group`	Group file containing groups names and group IDs
`/dev/tty*`	Login devices
`/etc/udb`	User validation file containing user control limits
`/etc/utmp`	User information file

SEE ALSO

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

`group`(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

`xadmin` – Manages user login accounts through a graphical user interface

SYNOPSIS

`xadmin` [`X11` *options*] [`-D` *t:d*] [`-p` *dir*] [`-u` *user*] [`-x` *dir*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `xadmin` command is a graphical user interface (GUI) that replaces the `nu`(8) command. It allows administrators to update or view the user database (UDB) on a UNICOS system. You can create, modify, or delete user accounts. All the functionality of `nu`(8) is available using `xadmin`.

The GUI is intended to be self-explanatory. Extensive online help is available from the main menu bar. The `help` button accesses `context help` or the `tutorial`. The `tutorial` button on the main menu bar lets you view the help topics either by clicking your mouse button on the selected help option or by paging through the text, using the scroll bar at the right of the tutorial screen. Help topics describe X11 Window System settings and the `xadmin` displays.

When invoking `xadmin`, the command line can include the following options:

`X11` *options*	Sets the X11 default values if the *options* argument is missing. Alternatively, you can specify any valid `DISPLAY` shell environment variables as *options*.
`-D` *t:d*	Sets message tracing and debugging levels. *t* is a trace message level integer value between 0 and 9 (inclusive). *d* is a debug message level integer value between 0 and 9 (inclusive). Generally, the higher the value you set, the more detailed will be the message and debugging information you receive.
`-p` *dir*	Sets the path name of the directory that contains the `udb`, `udb.public`, `acid`, `group` `passwd`, and `nu.cf60` files. The default directory is `/etc`.
`-u` *user*	Sets read-only mode. Specifies the user name or the user ID whose data will be displayed initially. Read-only mode also is invoked if you do not have access privilege to the UDB when you execute the `xadmin` command.
`-x` *dir*	Sets the path name of the directory that contains the files for the field computations and the prototypes. The default directory is `/etc/nulib`.

NOTES

The `xadmin` command traps interrupt signals (such as <`CONTROL-c`>) and does not end if you try to stop it in the middle of a critical section. Critical sections are primarily the updates of the UDB. A list of changes is recorded in a log file, usually `/usr/adm/xa.log`.

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

Active Category	**Action**
`system`, `secadm`	Modify all UDB fields.
`sysadm`	Modify all UDB fields, except sensitive fields.

If the `PRIV_SU` configuration option is enabled, the super user may modify all UDB fields.

CAUTIONS

Because any change to the `xadmin` or the `nu`(8) command executes as `root` user, a poorly written script potentially could make changes anywhere in the system.

WARNINGS

When you are experimenting with `xadmin`, you should use copies of the system UDB files from the `/etc` directory in a local directory, reducing risk to the UDB. For more information on creating test files, see the online `xadmin` help menu.

BUGS

The extensive use of shell scripts for doing sensitive duties (such as deleting and changing accounts) means that someone may be able to make `xadmin` fail in destructive ways without having access to the source code.

FILES

For the following files the default directory is `/etc`; you can change it by using the `-p` option.

`/etc/acid`	Account file
`/etc/group`	System group file
`/etc/nu.cf60`	Configuration file
`/etc/nulib/*.sh`	Shell scripts to perform the work
`/etc/nulib/nu.cf60`	Release default configuration file
`/etc/udb`	User validation file that contains user control limits
`/etc/udb.public`	Public version of the user database shell scripts to perform the work
`/usr/include/udb.h`	User database structure

SEE ALSO

`nu`(8), `udbgen`(8)

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

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

`acct`(5), `group`(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

`xdi` – Graphical user interface for X Window System based diagnostic applications

SYNOPSIS

All GigaRing based Cray Research systems:

`/etc/diag/xdi` [`-help`] [`-machine` *host-of-diagnostic-client*] [`-trace`] [*diagnostic-client-name*]

Cray system workstation (SWS):

`/opt/CYRIccmt/bin/xdi` [`-help`] [`-machine` *host-of-diagnostic-client*] [`-trace`] [*diagnostic-client-name*]

IMPLEMENTATION

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

DESCRIPTION

The `xdi` graphical user interface (GUI) makes available a generic X Window System interface for client/server diagnostic applications.

`xdi` can be started by using default or user-supplied command-line options. Default resources can be defined in the application-defaults file `/usr/lib/X11/app_defaults/XDi`.

The `xdi` command accepts the following options:

`-help` Specifies that the command-line synopsis should be written to `stderr`, after which `xdi` immediately exits.

`-machine` *host-of-diagnostic-client*
Specifies the machine on which the diagnostic client should be running.

`-trace`
Specifies that a trace file be created that will contain a variety of information reflecting the flow of execution.

diagnostic-client-name
Specifies the diagnostic client to which to connect.

To start `xdi` using default options, enter the following on the command line:

```
xdi
```

To start `xdi` using the command-line options, enter a command similar to the following:

```
xdi -m cool diagsio
```

A pop-up window is displayed that prompts you for the diagnostic client and the host machine. If you specified this information on the command line, or if default options are defined in the application-defaults file, the information is displayed in the appropriate input fields of the pop-up window (to allow you to modify the settings if necessary). Click on the `OK` button to initiate the connection to the diagnostic client.

After the appropriate information is gathered, `xdi` tries to connect to the specified diagnostic client. If the diagnostic client is not running, `xdi` attempts to start the client (not yet implemented).

When the `xdi` interface appears, text and data specific to the requested diagnostic client also appear.

To exit the `xdi` interface, click on the `Exit` option from the File pull-down menu (which is displayed by clicking on the `File` button in the menu bar at the top of the interface window).

NOTES

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

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

MENUS

`xdi` has four menus:

- `File`
- `Logs`
- `Connections`
- `Help`

The following are the options on the `File` menu:

`Open...`
: Brings up a file selection dialog that enables you to navigate the file system on which `xdi` is executing. From this dialog you can select a file whose contents will be displayed in a scrolled text window.

`Save` Saves the contents of all `xdi` regions and windows to the file `xdi.save`.

`Save As...`
: Saves the contents of all `xdi` regions and windows to the file name of your choice.

`Print` Prints the contents of all `xdi` regions and windows.

`Trace` Toggles whether or not the trace file facility is active.

`Exit` Exits `xdi` after displaying a confirmation window.

The following are the options on the `Logs` menu:

`Version`
: Displays a scrolled text window containing UNICOS source manager (USM) version information from `xdi` and the diagnostic client.

`Error` Displays a scrolled text window containing any error messages `xdi` needs you to see. When an error occurs, the keyboard bell is rung.

`Message`
: Displays a scrolled text window containing messages from the diagnostic client.

The following are the options on the `Connections` menu:

`Create...`
: Displays a window asking for information about which diagnostic client to connect to and on which machine that diagnostic client shoule be running. This menu item is active only when there are no current connections.

`Delete`
: Deletes the connection to the diagnostic client. This menu item is active only when an active connection exists.

The following are the options on the `Help` menu:

`Overview`
: Deferred implementation.

`Diagnostic`
: Requests help information from the diagnostic client. Because the diagnostic client knows what is being displayed by `xdi`, the help is typically context-sensitive.

`On_Item`
: Turns the mouse pointer into a question mark. Selecting a region of `xdi` using this pointer displays a scrolled text window that contains information about how `xdi` is treating that region.

`Using_Help`
: Deferred implementation.

`About_xdi`
: Deferred implementation.

LAYOUT

The `xdi` interface is divided into several regions. Each region is independent of the other regions. Any relationship between information in one region and information in another is defined by the diagnostic client. The regions are as follows:

Title bar
: Used by the diagnostic client to display short messages to the user.

Lists A, B, and C
: These lists can be independently configured in three different ways: (1) as a list that allows a single selection and immediately informs the diagnostic client; (2) as a list that allows a single selection without the immediate action; and (3) as a list that allows multiple selections. These configurations are completely controlled by the diagnostic client. For the latter two configurations, the selected information is sent to the diagnostic client when a button from the parameter region is pressed.

: The left-most list is *list A*. Just to the right of list A is *list B*. If *list C* is being used at all, it will be directly under list B, and list B and list C will be half the size of list A.

Typically, list A contains a list of actions that the diagnostic client can perform, and list B contains the devices or files on which to perform those actions.

Parameters
: This region can contain any number of parameter descriptions and entry fields, configured 8 or 16 to a page. These entry fields can be configured to accept a general character string, only numeric values, or a true/false toggle. The diagnostic client can, and usually does, place a variety of buttons across the bottom of the parameters. `xdi` does not receive information about what these buttons do, but you can try the `Help/Diagnostic` menu item for an explanation.

Output
: This is a scrolled output text area that is of arbitrary length. The visible size is 21 rows and 80 columns. The diagnostic client can place a variety of buttons across the bottom of the text area.

Command-line
: This region allows you to send an arbitrary character string to the diagnostic client. Some diagnostic clients treat the string as a command to be processed by the operating system in which it is running, but this is not the only purpose.

X WINDOW SYSTEM RESOURCES

The following X Window System resources apply to `xdi`:

Resource	Description
`diagClientMachine` (class DiagClientMachine)	Specifies the name of the machine on which the diagnostic client should run.
`diagClientName` (class DiagClientName)	Specifies the name of the diagnostic client to which to connect.
`help` (class Help)	If true, the command-line synopsis is displayed to `stderr`, and `xdi` immediately exits. The default is false.
`printCommand` (class PrintCommand)	Specifies the local print command to be used when the `File --> Print` menu item is chosen.
`trace` (class Trace)	If true, a trace file is produced as `xdi` executes. This feature is helpful for debugging purposes, but can create large files if left on for an extended period of time. The default is false.

ENVIRONMENT VARIABLES

The following environment variables apply to `xdi`:

Variable	Description
`DISPLAY`	Specifies to which X11 bit map display to attach.

`XAPPLRESDIR`	Specifies a directory in which to look for X Window System resource files in addition to the system-defined locations.
`XKEYSYMDB`	Specifies the location of an alternate keyboard symbol table. All X11R5 installations install a file named `XKeysymDB`, but this file does not always contain the newest definitions needed by a Motif application like `xdi`. The package that `xdi` was installed from should contain a file named `XKeysymDB` to be used with `xdi`.

FILES

`tracefile.*`	File that contains trace information.
`xdi.save`	Default file name to be used when the `File/Save` menu item is chosen.
`XDi`	Resource file for `xdi`.

SEE ALSO

`diagccmt`(8) for information on the diagnostic client for online I/O exercisers

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

NAME

`xdms` – Provides online disk maintenance and initialization routines for Cray GigaRing attached SCSI and fibre channel devices, which are driven by the `xdd` driver

SYNOPSIS

`/etc/xdms -a disk` [`-i` | `-u`] *iopath* | *iopath.unit*

`/etc/xdms -a info` *iopath.unit*

`/etc/xdms -a init` [`-m` *mode*] [`-A` *alternate path*] [`-M` *minor*] *iopath.unit*

`/etc/xdms -a init` [`-m` *mode*] [`-M` *minor*] *iopath*

`/etc/xdms -a init` [`-M` *minor*] [`-f` *filename*]

`/etc/xdms -a flaw -b blkno` *iopath.unit*[`-`*spindle*]

`/etc/xdms -a format -b block size` *iopath.unit*[`-`*spindle*]

`/etc/xdms -a readft` *iopath.unit*[`-`*spindle*]

`/etc/xdms -a read` [`-b` *blkno range* | `-C`] [`-p` *pattern*] [`-r` *request size*] [`-l` *loop*] [`-s` *seed*] [`-vz`] *iopath.unit*[`-`*spindle*]

`/etc/xdms -a write` [`-b` *blkno range* | `-C`] [`-m` *mode*] [`-p` *pattern*] [`-s` *seed*] [`-r` *request size*] [`-l` *loop*] [`-vz`] *iopath.unit*[`-`*spindle*]

`/etc/xdms -a surf` [`-b` *blkno range* | `-C`] [`-m` *mode*] [`-p` *pattern*] [`-s` *seed*] [`-r` *request size*] [`-l` *loop*] [`-vz`] *iopath.unit*[`-`*spindle*]

`/etc/xdms -a reconstruct` *iopath.unit*

`/etc/xdms -a disable` *iopath.unit*`-`*spindle*

`/etc/xdms -a scrub` *iopath.unit*

`/etc/xdms -a spinup` *iopath.unit*[`-`*spindle*]

`/etc/xdms -a spindown` *iopath.unit*[`-`*spindle*]

`/etc/xdms -a lducode -f filename` *iopath.unit*

`/etc/xdms -a ppage` [`-p` *pagenum*] `-r` *iopath.unit*

`/etc/xdms -a inquiry` *iopath.unit*

IMPLEMENTATION

Cray Research systems that support GigaRing based multipurpose nodes (MPNs) and fibre channel nodes (FCNs) running SCSI disks

DESCRIPTION

The `xdms` (eXtent Disk driver Maintenance System) command provides disk maintenance, initialization, and information gathering for SCSI or Fibre/SCSI devices attached to a multipurpose node (MPN) and the fibre channel node (FCN-1).

The command line is made up of the command to run (`xdms`), the `-a` option defining the action, and the desired options for the action, followed by the *iopath.unit* to be run to. If the action is to be run against an individual spindle of an array device, a spindle indicator may also be supplied to the *iopath.unit* field (for example, `020204.11-3`, which is ring 20, node 20, channel 4, unit 11, and spindle 3).

To execute `xdms`, a character device node must be set up in `/dev/xdd/xdiag`. This device node must be of major type `xdd` with a minor number of 0.

To run `xdms`, you must either have `diag` privilege set in the user database (UDB) or be running as `root`. In addition, to have permission to write to customer data areas, your UDB must also contain `guard` privilege or be running as `root`.

The `xdms` command accepts the following options:

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

`disk`	Generates output that defines the device mode, worldwide name (if a fibre channel device), vendor identification, model, serial number, I/O path defining the ring and node number given on the command line with the appropriate channel and unit number, and an alternate path if one can be determined. This default output may be used as input to the `init` action. See the EXAMPLES section for output from the `disk` action.
`info`	Provides output for a given disk device. The output consists of the Disk Unit Identification information of a Report Characteristics request as well as Disk Block information if the device has been initialized.
`init`	Defines the mode of access to be used by an FCN-1 device. MPN devices are automatically set to single-spindle devices so an `init` action is not required. However, for both FCN-1 and MPN devices, an `-M` option may be specified with a minor number to force the creation of a character special file in the `/dev/ddd` directory. See the description of the `-M` option. Device initialization may take various forms with the `init` action: (1) One device initialization where an *iopath* and *unit* identify the device to be initialized. This form supports all device modes. If a Redundant Arrays of Independent Disks (RAID) device mode is given, the *iopath.unit* must identify the parity or most significant bit in the spindle mask. This must also be the lowest unit in the subrack comprising the RAID device. A sufficient number of devices must fill the subrack to make up the rest of the array.

(2) Full SPN initialization where the *iopath* is given minus the *unit* field, indicating to `init` that all uninitialized devices are on the SPN. If the `-m` option is given with an argument of 1, ALL devices will be initialized as single-spindle devices. This form only supports device mode 01 (single-spindle drive).

(3) Initialization from an input file. This file must be in the form of the output of the `disk` action. The file will be read in and checked for validity. Each device is then initialized with the device mode. This action will overwrite currently set device modes. If a RAID device is to be initialized, each spindle of the device must be entered in order beginning with the parity or most significant spindle in the spindle mask.

Valid device modes are:

00	Uninitialized drive
01	Single-spindle drive
35	RAID-3 4+1 array
54	RAID-5 3+1
55	RAID-5 4+1
56	RAID-5 5+1
57	RAID-5 6+1
58	RAID-5 7+1
59	RAID-5 8+1

Note: Each RAID device requires that there be enough devices consecutively placed to complete the RAID configuration. Each device must be in the same subrack.

`flaw` Reassigns a failing block to the next available spare block in the spare blocking scheme. This action results in the loss of data to the target block. This action is typically only required on unrecoverable block errors because auto reassignment is done by the small computer system interface (SCSI) devices on recoverable error correction code (ECC) read errors. When flawing a block of an array, the spindle value must be supplied with the *iopath.unit*.

`format` Initiates the device formatting process. The block size will default to the current format size of the device. The `-b` option may be used to select a different block size. The only value currently accepted is 512, which indicates 512 Cray words or 4096 bytes. When formatting an array, the spindle value must be supplied with the *iopath.unit*.

`readft`	Reads and displays the defect list in block form and displays the defect type in bit mask form. If this action is specified to an array, the spindle value must be supplied with the *iopath.unit* field to indicate from which spindle to read the flaws.
`read`	Reads the specified block range given in the `-b` option or the CE blocks if a `-C` option is specified. If the `-p` *pattern* option is specified, the data read will be compared against that pattern. Use the `-r` option to select the size of each request made. If a request size is not given, the request size will be the optimal I/O value given in the disk unit block fields of the report characteristics request. The `-s` *seed* value is used with the `-p` *rand* option to generate consistent pseudo random data. The spindle value may be supplied to run to an individual spindle of an array.
`write`	Writes data across the block range given in the `-b` option or the CE blocks if `-C` is used. Data written is defined by the `-p` *pattern* option. The *mode* option is used by users with special privileges to access data areas otherwise restricted by general `xdms` users. The spindle value may be supplied to run to an individual spindle of an array.
`surf`	Performs a combination of the `write` then the `read` action and verifies the data read. The `surf` action imposes the same restrictions as the `write` action, which requires the `-m` option to bypass. The spindle value may be supplied to run to an individual spindle of an array.
`reconstruct`	Reinstates a downed spindle of an array and rewrites the data of each array block to the full complement of spindles in an array. This action is used when a failing spindle has been either replaced or fixed.
`disable`	Allows the manual disabling of a spindle in a disk array. This may be necessary where intermittent disk failures do not cause automatic disabling of the spindle.
`scrub`	Writes a pattern across an entire array to set data parity across the array.
`spinup`	Remotely spins up a spindle that has been spun down. The spindle value may be used to spin up an individual spindle of an array.
`spindown`	Remotely spins down a spindle. Typically this action is used to spin down a downed spindle of an array prior to removing it. The spindle value may be used to spin down an individual spindle of an array.
`lducode`	Reads in a file identified by the `-f` option into memory and then transfers it to the device specified by the *iopath.unit*. This file should contain the microcode that is to be used to update the microcode level of the disk drive. If the load microcode action is run on an array, each spindle of the array will be loaded with the new code.

`ppage`	Prints the SCSI mode pages from the device. If the mode page is a recognizable mode page, the output will define the values. Otherwise, the output will be in raw hexadecimal form. The `-r` option displays all pages in raw hexadecimal form. The `-p` option may be used to display individual mode pages. This option is currently available only for MPN devices.
`inquiry`	Prints the contents of the SCSI `inquiry` command. This option is currently available only for MPN devices.

`-A` Use with the `init` action to provide an alternate path. May be used when a device node is to be created. See the description of the `-M` option.

`-b` Use with the `read`, `write`, and `surf` actions to indicate the range of blocks to be read/written to. The block range is specified by giving a lower block value followed by a dash (-), followed by the upper block value. For example, `-b 5-10` executes on blocks 5 through 10.

Use with the `flaw` action to indicate which block to be assigned to an alternate block.

Use with the `format` action to indicate the desired block size to be used during formatting. The default is the current block size of the device.

`-C` May be used in place of the `-b` option on the `read`, `write`, and `surf` actions to indicate to run to the CE blocks only.

`-f` Where indicated, the `-f` option specifies the file name to be used for the appropriate action.

`-i` Use with the `disk` action to display only initialized devices.

`-l` Use with the `read`, `write`, and `surf` actions to indicate the desired pass count (the default is 1).

`-m` Use with the `init` action to indicate the device mode desired. The default device mode is 1 (single-spindle device).

Use with the `write` or `surf` action to allow a user to write customer data areas if an `unguard` option is specified. Only users with `perm guard` privilege in their user database (UDB) may allocate an `unguard` option (the default is `guard`).

`-M` Use with the `init` action to tell `xdms` to create a device node in `/dev/ddd` using the device characteristics obtained by the device. The argument given with this option should be a valid nonzero minor number. If the `-A` option is also specified, the device node will be created with an alternate path.

`-p` Use with the `read`, `write`, and `surf` actions to indicate the desired pattern to be used when testing. Valid options are zeros, ones, `hilo`, `peak`, `hole`, `bump`, `rand`, `seq`, and `addr`. The default is zeros, ones, `hilo`, `peak`, `hole`, `bump`, and `rand`. Use with the `ppage` action to identify the mode page (*pagenum*) to print.

`-r` Use with the `read`, `write`, and `surf` actions to indicate the desired size of each request to be issued. The default is the optimal IOU value given with the disk block information. Use with the `ppage` action to print mode pages in raw hexadecimal form.

`-s` Specifies the seed value to be used when generating random data. The default value is generated by the real time clock.

`-u` Use with the `disk` action to display only uninitialized devices.

`-v` Where indicated, runs `xdms` in verbose mode.

`-z` Where indicated, bypasses questions requesting a response to continue. The action will be run without prompting the user.

EXAMPLES

Example 1: Displays the disks attached to ring 20 and node 20. This information may be used as input to the `init` action.

```
# ./xdms -a disk 020200
*
*mode    wwname          model             s/n     iopath.unit   altpath
*
 01 1000002037000061  ST15150FC         88071     020202.1       ------
 01 1000002037000062  ST15150FC         88072     020202.2       ------
 01 1000002037000063  ST15150FC         88073     020202.3       ------
 01 1000002037000064  ST15150FC         88074     020202.4       ------
 01 1000002037000065  ST15150FC         88075     020202.5       ------
 01 ----------------  ST15150FC         88076     020201.1       ------
```

Note: MPN devices are identified by the "--" sequence in the `wwname` field, while FCN-1 devices are shown with valid hexadecimal values in the `wwname` field. The `altpath` field will have a valid path name if an alternate path can be determined. Currently, alternate paths are only determined by having valid channels on both port A and port B for a given device node.

Example 2: Shows a typical `info` action output for ring 20, node 02, channel 0, and unit 4.

```
# ./xdms -a info 020020.4
*
* Disk Information for   020020.4
*
* Mode: 01      Device Ordinal: FFFF      Channel PrtA: 00  PrtB: ff
* Disk Unit State: FF    Spindle Mask: 0000    Unit: 004    LUN: 000
* Interface Type: 02     Command Protocol: 01     Vendor ID: SEAGATE
* Model Number: ST151150C              Serial Number: 12345678
* Firmware Revision: 2500       World Wide Name:  ---------------
*
* Block Size: 512       ECC Field Length: 24   Optimal IOU: 10
* Number of Blocks: 250   CE Blocks: 240-249
```

Example 3: Initializes the device defined by ring number 020, node number 02, channel 4, and unit 21 as a (default) single-spindle device.

```
# ./xdms -a init 020024.21
```

Example 4: Initializes the device defined by ring number 020, node number 02, channel 4, and unit 21 as a RAID-3 array defined by the `-m` option. Units 22, 23, 24, and 25 make up the other spindles of the array. Unit 21 would be the most significant bit of the spindle mask 020 (octal) while unit 25 is the least significant bit 001 (octal). The full mask would be 037 (octal).

```
# ./xdms -a init -m 34 020024.21
```

Example 5: Initializes the device defined by ring number 020, node number 02, channel 4, and unit 21 as a RAID-3 array defined by the `-m` option. The `-A` option defines the alternate path to the physical array.

```
# ./xdms -a init -m 34 -A 020042 020024.21
```

Example 6: Initializes all of the devices on ring number 020 and node number 02 that have a current mode of 00 (uninitialized) as single-spindle devices. All other devices will be unaltered.

```
# ./xdms -a init 020020
```

Example 7: Initializes all of the devices on ring number 020 and node number 02 as 01 (single-spindle device) no matter what the current mode state setting. The `-m` option of 01 forces this state on ALL devices.

```
# ./xdms -a init -m 01 020020
```

Example 8: Creates a file named `myfile` using the `info` action. The file may then be updated by the user to select the desired device modes and alternate path selections. This file is then given as input to the `init` action with the `-f` option.

```
# ./xdms -a info 020020 > myfile
# ./xdms -a init -f myfile
```

SEE ALSO

`udbgen`(8) for information on setting up permbits in the UDB

NAME

`xfddimap` – X Window System utility to draw an FDDI ring map

SYNOPSIS

`/etc/xfddimap` [`-d`] [`-p` *port*] [`-i` *infc*] [`-f` *font*]

IMPLEMENTATION

Cray PVP systems with I/O subsystem model E

DESCRIPTION

The `xfddimap` utility transmits FDDI station management (SMT) frames onto an FDDI ring from a Cray Research system that is directly attached to that same FDDI ring, to draw a logical ring map of the FDDI network. Users may also monitor some of the FDDI counters within the FDDI stations connected to the ring and gather information about the stations on the network.

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

The command-line options are as follows:

`-d`	Enables debug mode; verbose output.
`-p` *port*	Sets the UDP port number to which socket requests are sent. Default is `3000`. See `smtd`(8).
`-i` *infc*	Sets the interface ordinal of the FDDI interface to use when sending requests on the FDDI ring. Default is 0.
`-f` *font*	Sets the X11 font to the specified font type.

The MAC Address File

You can create a file that contains the IEEE 48-bit media access protocol (MAC) addresses of the FDDI stations (similar to the `/etc/hosts` file) to give names to each of the FDDI stations on the ring. This file is called `/etc/ethers` and is standard on most UNIX-based systems. The format of the file is as follows:

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

You can place comment lines anywhere in the file if the comments start in column 1 of the line. Comments are designated by a # character in column 1. Leading zeros within each byte are not required, but they do make it easier for the users to read.

The `xfddimap` utility consults this file to associate a physical MAC address to a station (or host) name so that names and addresses appear in the ring map.

Logical Ring Map Display

After it is drawn, the logical ring map consists of several rectangles, each representing a physical FDDI station on the ring. Inside each rectangle three items of information appear. The name of the station as found in the `/etc/ethers` file appears first. Next is the station's CFM state (`WRAPPED` or `THRU`). Finally, the station's 48-bit MAC address is displayed in IEEE (Canonical) form. The arrow that is drawn between the rectangles indicates the direction of token flow; therefore, stations to the right are downstream from the station to their left.

User Actions

Users can invoke some simple actions by placing the cursor inside one of the stations in the ring map and pressing any mouse button. Pressing a button causes `xfddimap` to draw an additional window on the screen, which displays the MAC counters from within the FDDI station. In the tty window from which the program was started, a summary of station information is displayed. This information consists of configuration station information (CSIF), operational station information (OSIF), and neighbor information (NIF). The information should be self-explanatory; however, users should read the station management (SMT) ANSI document (see the SEE ALSO section) for a more detailed explanation.

FILES

`/etc/ethers`	File containing MAC addresses of all `FDDI` stations on an `FDDI` ring

SEE ALSO

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

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

The ANSI documents for FDDI:

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

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

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

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

Other documents related to FDDI:

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

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

NAME

xmppview – Displays Cray MPP system activity through a graphic user interface

SYNOPSIS

/usr/bin/xmppview [-*toolkitoption* ...]

IMPLEMENTATION

Cray MPP systems

DESCRIPTION

The xmppview command graphically displays system activity of a Cray Research massively parallel processing (MPP) system. xmppview has all the functionality of the mppview(8) command. The displays show changes in MPP system utilization in approximate real time. System administrators can use this utility on a continuous basis to monitor MPP system activity and to request specific reports. System users can check resource availability before running jobs on the MPP system and see the status of jobs already initiated.

The xmppview command accepts most of the standard X Toolkit command-line options, [-*toolkitoption* ...]. See the X man page for a complete list of available options. Examples of common options include the following:

```
-display <display>
-geometry <geometry>
-fg <color>
-bg <color>
-xrm <resource string>
```

The xmppview utility is implemented with the standard Motif widget set, so you can modify the appearance of xmppview to suit your preferences by using the X resource database. For general information about X resources, see the xrdb(1X) and X(7X) man pages. The following are the default fallback resources for xmppview:

```
*background: LightSteelBlue
*XmFrame.XmDrawingArea*background: #305C7D
*labelLineForm*background: #B7C2E5
*labelLineForm*foreground: Blue
*helpLabel.foreground: Blue
```

Before you start the xmppview utility, check the following items:

- Set your DISPLAY environment variable to your workstation. For more information, see the X(7X) man page.
- Verify that the machine on which you want to run xmppview is in the list of valid host names allowed to connect to your X server. For more information, see the xhost(1X) man page.

- Set up your `.rhosts` file on the remote Cray MPP hosts on which you plan to monitor system activity. For more information, see the `rhosts`(5) man page.

To reduce work load on the Cray Research system, you can use a SPARC workstation or server to execute `xmppview`. In order to bring up the `xmppview` window, you must have your workstation environment set up to use the X Window System (X11) and to connect to the host system you want to monitor.

A tutorial is available through the `help` pull-down menu. You can navigate this tutorial by using the scroll bar at the right side of the window. The tutorial covers the following subjects:

```
    * Getting Started
      - Uses and users
      - Running xmppview
      - A quick tour
      - Error messages
    * Connecting to a Host
    * Viewing the 3D Animated Torus Display
      - Viewing different attributes
      - Using the fill options
      - Changing the view orientation
      - Options
    * Viewing Dynamic Text Reports
    * Customization
```

NOTES

Invoking `xmppview` from a mainframe or workstation launches the `mppview` command on the designated host Cray Research system with an MPP system, unless the `XMPPVIEW_HOSTCMD` environment variable is set other than to the default.

CAUTIONS

(`xmppview` running on CRAY Y-MP E, CRAY Y-MP M90, or CRAY C90 systems only)
Although the `xmppview` utility's output can be displayed on most X Window System servers, displaying it on an operator workstation (OWS) or other SPARC workstations running an OpenWindows version prior to 3.0 is not recommended, because of known adverse interaction between current Motif libraries and older versions of OpenWindows.

This problem does not exist when running `xmppview` directly from the OWS (because the SPARC `xmppview` binary is built with older Motif libraries).

ENVIRONMENT VARIABLES

When starting up, the `xmppview` utility looks for the following optional environment variables:

`XMPPVIEW_INITHOST`

The value of this variable should be the name of the host machine that you want `xmppview` to connect to upon startup of the command. By default, when running `xmppview` on a CRAY Y-MP E, CRAY Y-MP M90, or CRAY C90 system, `xmppview` will try to connect to the local host (that is, the machine on which `xmppview` is running).

The literal string `<Local_Host>` is recognized by `xmppview` to mean the machine on which `xmppview` is running.

`XMPPVIEW_HOSTLIST`

The value of this variable should be a list of host names, separated by blank spaces, that you want to appear first in the item list of the Host Setup dialog box. Host names in this list will be available for simple point and click operations in that dialog box.

`XMPPVIEW_QUERYINT`

The value of this variable should be an integer from 1 through 9. The integer value represents the number of seconds between updates. The value controls the rate at which `xmppview` queries for information from the Cray MPP system and updates the screen display.

If you are displaying on a slow `X` terminal (perhaps across a serial line connection), you may want to set this variable to a higher value. The default is 3.

`XMPPVIEW_HOSTCMD`

The value of this variable should be a command line for the `mppview` command or a similar program that the `xmppview` command can query for its information. The default is `/usr/bin/mppview -t`.

MESSAGES

To the Terminal

Error messages sent to your terminal have the following meanings:

```
Warning: cannot open display
```

This message indicates that you have attempted to execute `xmppview` with the display set to a host that is not running an `X` server. Typically, this occurs as a result of running the program on a machine with these conditions:

- no `DISPLAY` environment variable set, and
- no `-display` command-line option specified.

For information on how to correct these conditions, see the `X`(7X) man page.

```
Xlib: connection to "<workstation>:<display>.<screen>" refused by server
Xlib: Client is not authorized to connect to Server
```

`Warning: cannot open display`
: These messages indicate that you have attempted to execute `xmppview` with the display set to an `X` server that is not set up to accept connections from the machine on which you are running `xmppview`. For information on how to add host names to the list of machines allowed to make connections to the `X` server, see the `xhost`(1X) man page.

To a Pop-up Window

The message `Error:  Unable to connect to MPP`, when found in a pop-up window, is accompanied by an explanation, such as the following:

`Can’t talk to samdaemon from <host>`
: This message indicates that a current version of the `samdaemon` is not running on the currently selected CRAY Y-MP E, CRAY Y-MP M90, or CRAY C90 host system with the Cray MPP system. Check with your system administrator to resolve this error.

`No MPP system present`
: This message indicates that `xmppview` has attempted to access a host system that is not running UNICOS MAX software.

SEE ALSO

`csam`(8), `mppview`(8), `samdaemon`(8), `xsam`(8)

UNICOS Resource Administration, Cray Research publication SG–2302

NAME

`xsam` – Displays graphic data about system activity

SYNOPSIS

`/usr/bin/xsam` [*X11-options*] [`-f` *script_file*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `xsam` facility is a graphics tool that monitors, in real time, the activity of a Cray Research system. It manages a set of displays that show data received from the system activity monitoring daemon, `samdaemon`(8). The displays are X Window System Version 11 windows; this man page assumes that you are familiar with the X Window System.

This description of `xsam` is only an introduction; for a more detailed description and lists of available X Window System options, see *UNICOS Resource Administration*, Cray Research publication SG–2302.

The `xsam` facility accepts the following options:

X11-options	Regular X11 options to control X11 defaults.
`-f` *script_file*	Name of a file that contains valid `xsam` commands. These commands are executed before control is passed to the calling user.

The `xsam` facility creates six types of display windows, as follows:

Window Type	**Description**
Console window	Controls all aspects of `xsam`. You can use it to request other `xsam` displays. It also contains a message section that displays all error and informational messages.
Configuration window	Shows aspects of the hardware and software configuration of the target system.
Graph window	Allows you to monitor various kernel counters over a long period of time. Possible graphs include CPU utilization, I/O statistics, and system call usage.
Map window	Displays a representation of memory. Within the map are several columns of boxes, each with its own name and each corresponding to a process in memory. The size of each box is proportional to the size of the process in main memory. As the process moves or grows, the box moves and grows. Command options allow you to display the process ID, view a subsection of memory, and view a snapshot of any process.

Snapshot window	Displays information about a specific process in text form. The data is extracted from the kernel's process structure for the target process. You can request a snapshot from a memory map or directly by using the snapshot command. The snapshot window contains several sets of statistics, including user information (such as the size and address of the process and user and process flags), scheduling parameters, and an area devoted to user identification.
Device I/O window	Displays device configuration and I/O performance on a system wide level, as well as on an individual device level. Possible device type are *disks* and *logical devices*.

SEE ALSO

`csam`(8), `sam`(8), `samdaemon`(8)

UNICOS Resource Administration, Cray Research publication SG–2302

NAME

`xtpldr` – Manages selected autoloader operations

SYNOPSIS

`xtpldr` [`-t` *trace_file*]

IMPLEMENTATION

Cray PVP systems

DESCRIPTION

The `xtpldr` command allows you to enter tape cartridges into the domain of an autoloader, eject tape cartridges from the domain of an autoloader, and query autoloaders for volume serial number (VSN) information.

The `xtpldr` command accepts the following option:

`-t` *trace_file* Specifies the file into which trace information is written. If you experience a problem, you can try to reproduce it by reissuing the `xtpldr` command with the `-t` option specified. The resulting trace information may be helpful to your support staff in resolving the problem.

The `xtpldr` command brings up a graphical user interface with the main window having the following components:

Main Window	**Description**
Menu bar	Consists of three buttons:

Button	**Description**
`exit`	Exits the `xtpldr` program.
`Cray Research, Inc.` (logo)	Activates a pop-up window that displays the following `xtpldr` information: • Version number • Name of computer system on which `xtpldr` is executing • UNICOS release that the host computer is executing • Compilation date • Copyright statement
`help`	Activates a pop-up window that displays `xtpldr` help information.

Instruction/message line	Displays messages that show you completion status for the operation you have selected or instructions for the next step.
`Available autoloaders`	Shows you the autoloaders that meet the following criteria: • Are configured up (`UP`) in the tape configuration. • Are connected to the Cray Research system through a network that uses the TCP/IP communication protocol.

Selecting Autoloader and Action

Before you can perform any autoloader operations, you must select a Storage Technology Corporation (StorageTek), IBM, or EMASS autoloader and an action.

1. Select an autoloader from the `available autoloaders` list box.

   To make a selection, position the mouse pointer over one of the entries listed and press the left mouse button.

   If the list consists only of a `retry` entry, your site supports autoloaders that satisfied the criteria specified, but they are not configured `UP`. When you select `retry`, the tape daemon re-examines the tape configuration. If the state of one or more autoloaders has changed to `UP`, these autoloaders replace the `retry` entry. If the state of the autoloaders has not changed, the `retry` entry is displayed.

2. Click on one of the following action push buttons:

Push Button	**Description**
`enter`	Enters tape cartridges into the domain of the selected autoloader.
`eject`	Ejects tape cartridges from the domain of the selected autoloader.
`query`	Obtains information from an autoloader concerning tape cartridge VSNs.

   After you select an autoloader, the `xtpldr` window is subdivided into sets of left-handed and right-handed panes. The autoloader and action selections are part of a set of left-handed panes into which you enter information. The right-handed panes list your `enter`, `eject`, and `query` VSN selections. For each action, you can erase your selections and restart the process by clicking the `clear vsn list` button at the bottom of the right-hand pane.

   During autoloader operations, various pop-up windows provide status information. To make a pop-up window disappear, click the `dismiss` button.

3. Follow the Entering Tape Cartridges procedure that pertains to your action selection. The instructions in instruction/message line will outline the basic steps in the procedure.

Entering Tape Cartridges

After you have chosen an autoloader and selected `enter`, select the procedure from the following table that applies to your StorageTek or IBM autoloader.

Autoloader	**Procedure**
StorageTek	To enter tape cartridges into the domain of a StorageTek autoloader, perform the following steps:

1. Click on the `cap info` button. Each Cartridge Access Port (CAP) holds tape cartridge storage cells and is used to enter tape cartridges into the domain of a Library Storage Modules (LSM) without opening the access door in the LSM outer wall. The door contains one or more CAPs.

2. Select a CAP from the `enter/eject stations in loader` *name* list box in the `ees information` pop-up window. For each available CAP, the list contains its address, capacity, mode, state, and status, as shown in the following example:

   ```
   0, 0,0  21 MANUAL   ON_LINE     STATUS_CAP_AVAILABLE
   ```

3. Enter the address of the selected CAP in `enter the cap id - a,l,c` input line below the CAP listing. The address is made up of the Automated Cartridge System (ACS) number, LSM number, and CAP number.

   After you enter a CAP, its address appears in the `cap info` button, and the instruction in the instruction/message line reads:

   ```
   cap selected: press ok to start enter
   ```

4. Click on the `ok` button below the `enter/eject stations in loader` *name* list box.

   - If the selected CAP is in `MANUAL` mode, the CAP is unlocked so that you can open it and put tape cartridges in it. After you close it, the autoloader robotics stores the cartridges in the LSM slots.
   - If the CAP is in `AUTOMATIC` mode, follow the instructions in the StorageTek documentation.

     The message in the instructions/message line reads:

     ```
     enter cartridges in the ees
     ```

5. Enter the tape cartridges in the enter/eject station (ees).

6. See the `vsns entered from loader` *name* list box in the `cartridge information` pop-up window for status information on each individual operation. For each cartridge, the display shows the CAP, VSN, and status, as shown in the following example:

   ```
   0, 0,0  000620  STATUS_SUCCESS
   ```

7. Click on the `vsns completed` push button when you are done entering tape cartridges.

8. Note the termination message in the instructions/message line:

```
no cartridges were in ees
```

IBM	Enter the tape cartridges in the enter/eject station (ees). The IBM autoloader server is not involved in this operation, and status information is not available.
EMASS	To enter tape cartridges into the domain of an EMASS autoloader, perform the following steps:

1. Select an archive from the `archives queried in loader` *name* list box in the `archive information` pop-up window.
2. Select a media class from the `media classes queried in loader` *name* list box in the `media class information` pop-up window.
3. Read the instructions in the instruction/message line:

   ```
   enter: single vsns for logical import
   ```

   The `vsn` and `filename` buttons, which appear opaque, are not available.
4. Enter the VSNs one at a time in the `information to be entered` input area.

   After you enter a VSN, it is displayed in the right-hand pane called `vsns to be entered`.
5. Click on the `vsns completed` push button to instruct the autoloader to add the VSNs to the internal list of VSNs associated with the specified archive and media class.
6. Check the `vsns entered into loader` *name* list box in the `cartridge information` pop-up window for status on each VSN that is logically imported. For each cartridge, the display shows its VSN and status, as shown in the following example:

   ```
   000002  volume_logically imported
   ```
7. Import the VSNs into the archive from the archive controlling display on the system that executes the autoloader server. For instructions, see your EMASS documentation.

Ejecting Tape Cartridges

After you have chosen an autoloader and selected `eject`, select the procedure from the following table that applies to your StorageTek or IBM autoloader.

Autoloader	**Procedure**
StorageTek	To eject tape cartridges from the domain of a StorageTek autoloader, perform the following steps:

1. Click on the `cap info` button.

2. Select a CAP from the `enter/eject stations in loader` *name* list box in the `ees information` pop-up window. For each available CAP, the list contains its address, capacity, mode, state, and status, as shown in the following example:

   ```
   0, 0,0  21 MANUAL   ON_LINE     STATUS_CAP_AVAILABLE
   ```

3. Enter the address of the selected CAP in the `enter the cap id - a,l,c` input line below the CAP listing.

   After you select a CAP, its address appears in the `cap info` button, and the instruction in the instruction/message line reads:

   ```
   select: select eject mode
   ```

4. Click on the `ok` button at the bottom of the `ees information` pop-up window.
5. Select the `single vsn` or `vsn ranges` button.
6. Select the `vsn` or `filename` button.
7. Enter the VSNs in the `information to be entered` input area using one of the following formats:

   - Single VSNs or VSN ranges for the `vsn` selection
   - Name of a file (80 characters or less) that contains VSN information in the format documented for the `-v` option of the `tpmnt`(1) command

   After you enter a VSN, it is displayed in the right-hand pane called `vsns to be ejected`.

8. Select the `vsns completed` push button.

   The autoloader robotics deposit the cartridges in the CAP you specified, and the status message in the instructions/message line reads:

   ```
   autoloader ejecting cartridges.
   ```

9. Remove the cartridges from the CAP and close the CAP door.
10. See the `vsns ejected from loader` *name* list box in the `cartridge information` pop-up window for status information on each individual eject operation. After you have emptied the CAP and closed the CAP door, status information is displayed, as shown in the following example:

    ```
    0, 0,0  000629  STATUS_SUCCESS
    ```

11. Note the termination message in the instructions/message line:

    ```
    final server reply received: 1
    ```

The number 1 in this message indicates that your cartridges have been removed from the CAP. If your list of VSNs exceeds the CAP capacity, the autoloader refills the CAP as many times as necessary to complete your request. The number increases with each completed refill.

IBM

To eject tape cartridges from the domain of an IBM autoloader, perform the following steps:

1. Enter a station identifier by clicking one of the following push buttons:

Station Identifier	**Procedure**
`convenience`	Selects the convenience enter/eject station of the IBM autoloader to deposit the ejected tape cartridges. The optional IBM 3494 station handles 10 or fewer tape cartridges.
`high-capacity`	Selects the high-capacity enter/eject station of the IBM autoloader to deposit the ejected tape cartridges. The IBM 3494 station handles 40 or fewer tape cartridges. Remove the tape cartridges after you have paused the autoloader and opened the door.

2. Enter the VSNs in the `information to be entered` input area using one of the following formats:

   - Single VSNs for the `vsn` selection
   - Name of a file (80 characters or less) that contains VSN information in the format documented for the `-v` option of the `tpmnt`(1) command

   After you enter a VSN, it is displayed in the right-hand pane called `vsns to be ejected`.

3. Select the `vsns completed` push button.

   The autoloader robotics deposit the cartridges in the enter/eject station you specified.

4. See the `vsns ejected from loader` *name* list box in the `cartridge information` pop-up window for immediate status information on each individual eject operation. For each cartridge, the display shows its VSN and status, as shown in the following example:

```
003600  volume ejected
```

5. Remove the cartridges and close the door.

6. Note the termination message in the instructions/message line:

```
final server reply received: 1
```

The number 1 in this message indicates that your cartridges have been removed from the CAP. If your list of VSNs exceeds the CAP capacity, the autoloader refills the CAP as many times as necessary to complete your request. The number increases with each completed refill.

EMASS — To eject tape cartridges from the domain of an EMASS autoloader, perform the following steps:

1. Select the `vsn` or `filename` button.

   You can alternate between the `vsn` and `filename` buttons without loss of previously entered information; alternating clears the information that remains in the input area in which you enter the information.

2. Enter the VSNs in the `information to be entered` input area using one of the following formats:

   - Single VSNs for the `vsn` selection
   - Name of a file (80 characters or less) that contains VSN information in the format in the `-v` option of the `tpmnt`(1) command

   After you enter a VSN, it is displayed in the right-hand pane called `vsns to be ejected`.

3. Select the `vsns completed` push button to logically export the specified VSNs.

4. Check the `vsns ejected from loader` *name* list box in the `cartridge information` pop-up window for status information on each individual logical export operation. For each cartridge, the display shows its VSN and status, as shown in the following example:

```
000003  volume_logically exported
```

5. Export the VSNs in the archive from the archive controlling display on the system that executes the autoloader server. For instructions, see your EMASS documentation.

Querying Autoloaders

After you have chosen an autoloader and selected `query`, select the procedure from the following table that applies to your StorageTek or IBM autoloader.

Autoloader — **Procedure**

StorageTek — To obtain information from a StorageTek autoloader concerning tape cartridge VSNs, perform the following steps:

1. Select the `vsn` or `filename` button.

2. Enter the VSNs in the `information to be entered` input area using one of the following formats:

   - Single VSNs for the `vsn` selection

- Name of a file (80 characters or less) that contains VSN information in the format documented for the `-v` option of the `tpmnt`(1) command

After you enter a VSN, it is displayed in the right-hand pane called `vsns to be queried`.

3. After you have entered all the VSNs that you want to query, select the `vsns completed` push button.
4. Check the `vsns queried in loader` *name* list box in the `cartridge information` pop-up window for status information about the specified VSNs. For each cartridge, the display shows the slot address, VSN, and status, as shown in the following example:

```
0, 0,15, 9, 0  000621 volume_in_loader: home
```

IBM	The instructions for the StorageTek autoloader are identical to the instructions for the IBM autoloader. The IBM autoloader server does not return the slot address of the tape cartridges.
EMASS	The instructions for the StorageTek autoloader are identical to the instructions for the EMASS autoloader. The EMASS autoloader server does not return the slot address of the tape cartridges.

EXIT STATUS

If `xtpldr` fails, an exit status code in the range of 1 through 255 is returned; where possible, the number is normalized to the last three digits. Exit status values are documented in the *Tape Subsystem User's Guide,* Cray Research publication SG–2051.

SEE ALSO

`tpmnt`(1), `tpquery`(1)

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

NAME

`ypinit` – Builds and installs the network information service (NIS) database

SYNOPSIS

```
/etc/yp/ypinit -m
/etc/yp/ypinit -s master_name
```

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `ypinit` command sets up an NIS database on a network information service (NIS) server. You can use it to set up a master or a slave server. It asks a few, self-explanatory questions, and reports success or failure to the terminal.

The `ypinit` command accepts the following options:

`-m` Indicates that the local host is the NIS master.

`-s` Sets up a slave database.

master_name
Specifies the host name of the NIS server (either the master server for all of the maps or a server on which the database is current and stable).

Cray Research currently supports the NIS distribution of only the following databases (master or slave server):

- `group`
- `passwd`
- `publickey`
- `ypservers`

It sets up a master server such that the server is master to all maps in the database. Later, you can change the association of maps to masters. All databases are built from scratch, either from information available to the program at run time, or from the ASCII database files in the `/etc` directory. All such files should be in their traditional form, rather than the abbreviated form used on client machines.

`ypinit` sets up an NIS database on a slave server by copying an existing database from a running server.

See the `ypfiles`(5) and `ypserv`(8) man pages for an overview of the NIS.

SEE ALSO

`makedbm`(8), `yppush`(8), `ypserv`(8), `ypxfr`(8)

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

NAME

`yppasswdd` – Handles password change requests from `yppasswd`(1)

SYNOPSIS

`/etc/yppasswdd` *file* [`-m` *arg1 arg2* ...]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `yppasswdd` program is a server that handles password change requests from `yppasswd`(1). It changes a password entry in the specified *file*, which is assumed to be in the format of the `/etc/passwd` file (see `passwd`(5)). An entry in the *file* is changed only if the password presented by `yppasswd`(1) matches the encrypted password of that entry.

After *file* is modified, if the `-m` option is specified, a `make`(1) is performed in the `/etc/yp` directory. Any arguments that follow the option are passed to `make`.

The `yppasswdd` command accepts the following options:

file	Specifies the file that contains the password entry to be changed.
`-m` *arg1 arg2* ...	Specifies that a `make` function will be performed; the accompanying arguments are passed to the `make` command. This server is not run by default, and it cannot be started from `inetd`(8). If you want to enable remote password updating for network information service (NIS), place an entry for `yppasswdd` in the `/etc/rc` script (see `brc`(8)) of the host serving as the master for the NIS.

The `yppasswdd` program catches the `SIGHUP` signal and reregisters itself with `portmap`(8) when it receives the signal. This enables `yppasswdd` to continue running properly when `portmap`(8) must be restarted.

EXAMPLES

If the NIS password file is stored as `/etc/yp/src/passwd`, to propagate password changes immediately, invoke the server as follows:

```
/etc/yppasswdd /etc/yp/src/passwd -m -f yp.mk passwd
```

In this case, `src` is the NIS domain name.

FILES

`/etc/yp/yp.mk`	Make file
`/etc/passwd`	Password file

SEE ALSO

`brc`(8), `portmap`(8)

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

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

NAME

`yppoll` – Finds network information service (NIS) map version at NIS server host

SYNOPSIS

`/etc/yp/yppoll` [`-h` *host*] [`-d` *domain*] *map*

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `yppoll` command asks a `ypserv`(8) process what the order number is, and which host is the master network information service (NIS) server for the specified *map*. If the server is a V.1 YP protocol server, `yppoll` uses the older protocol to communicate with it. In this case, it also uses the older diagnostic messages in case of failure.

The `yppoll` command accepts the following options:

`-h` *host*	Asks the `ypserv` process at the specified host about the map parameters. If the host is not specified, the NIS server for the local host is used (that is, the default host is the one returned by the `ypwhich`(1) command).
`-d` *domain*	Specifies a domain to be used instead of the default domain.
map	Specifies the name of the map being queried.

SEE ALSO

`ypserv`(8)

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

NAME

`yppush` – Forces the propagation of a changed network information services (NIS) map

SYNOPSIS

`/etc/yp/yppush` [`-d` *domain*] [`-v`] *map*

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `yppush` command copies a new version of an NIS map from the master NIS server to the slave NIS servers. It is usually run only on the master NIS server by the makefile in `/etc/yp/` after the master data bases are changed. It first constructs a list of NIS server hosts by reading the NIS map `ypservers` within the domain. Keys within the `ypservers` map are the ASCII names of the machines on which the NIS servers run.

A transfer map request is sent to the NIS server at each host, along with the information that the transfer agent needs (the program that actually moves the map) to call back the `yppush`. When the attempt has completed (successfully or not), and the transfer agent has sent `yppush` a status message, the results may be printed to `stdout`. Messages are also printed when a transfer is not possible; for instance, when the request message is undeliverable, or when the timeout period on responses has expired.

See `ypfiles`(5) and `ypserv`(8) for an overview of the NIS.

The `yppush` command accepts the following options:

`-d` *domain*	Specifies a domain.
`-v`	Specifies verbose mode. This causes a message to be printed for each response when each server is called. If you omit this option, only error messages are printed.
map	Specifies the name of the map to be copied.

BUGS

In version 2 NIS, the transfer agent is `ypxfr`(8), which is started by the `ypserv` program. If `yppush` detects that it is speaking to a version 1 NIS protocol server, it uses the older protocol, sending a version 1 `YPPROC_GET` request and issuing a message to that effect. Unfortunately, you cannot know if or when the map transfer is performed for version 1 servers. `yppush` prints a message saying that a version 1 message has been sent. You should check to see that the transfer has actually occurred.

FILES

`/etc/yp/`*domainname*`/ypservers`   NIS map

SEE ALSO

`ypserv`(8), `ypxfr`(8)

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

NAME

`ypserv`, `ypbind` – Provides network information service (NIS) server and binder processes

SYNOPSIS

`/etc/ypserv`
`/etc/ypbind` [`-v`] [`-h` *hostname* [*hostname*] ...]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The network information service (NIS) function provides a simple network look-up service that consists of databases and processes. The databases are `dbm`(3C) files in a directory tree rooted at `/etc/yp`, and they are described in `ypfiles`(5). The processes are `/etc/ypserv`, the NIS database look-up server, and `/etc/ypbind`, the NIS binder. The program interface to NIS is described in `ypclnt`(3C). Administrative tools are described in the `yppush`(8), `ypxfr`(8), `yppoll`(8), `ypwhich`(1), and `ypset`(8) man pages. Tools that are used to see the contents of NIS maps are described in the `ypcat`(1) and `ypmatch`(1) man pages. Database generation and maintenance tools are described on the `makedbm`(8) and `ypinit`(8) man pages.

Both `ypserv` and `ypbind` are daemon processes which are typically activated at system start-up time by the `rc`(8) script. The `ypserv` command runs only on NIS server machines that have a complete NIS database. The `ypbind` command runs on all machines that use NIS services, both NIS servers and clients.

The `ypserv` daemon looks up information in its local database of NIS maps. Communication to and from `ypserv` is accomplished by Remote Procedure Call (RPC) calls. Look-up functions are described in the `ypclnt`(3C) man page, and they are supplied as C-callable functions in `libc`. The following look-up functions are performed on a specified map within an NIS domain:

`Get_all`	Ships the entire map to the requester as the response to one RPC request
`Get_first`	Returns the first key-value pair from the map
`Get_next`	Enumerates all but the first key-value pair from the map
`Match`	Takes a key, and it returns the associated value

The `Get_order_number` and `Get_master_name` functions supply information about the map, rather than map entries. In fact, both order number and master name exist in the map as key-value pairs, but the server does not return either through the normal look-up functions. (However, if you examine the map by using the `makedbm`(8) command, they are visible.)

`Do_you_serve_this_domain?`, `Transfer_map`, and `Reinitialize_internal_state` functions are also used within the NIS subsystem, but they are not of general interest to NIS clients.

The `ypbind` program remembers information that lets client processes on one node communicate with a `ypserv` process.

This program accepts the following options:

`-v` Specifies that `ypbind` will not put itself in background mode and that it will print debugging messages to standard output. If you create the `/etc/yp/ypserv.log` file, debugging messages are also written to this file.

`-h` *hostname* [*hostname*] . . .
Specifies a list of hosts that are NIS servers for the current NIS domain to attempt to bind. The default is for the Cray Research system to bind to itself, because the Cray Research system does not support broadcast mode over its networks.

The `ypbind` process must run on every machine that has NIS client processes; `ypserv` may not be running on the same node, but it must be running somewhere on the network.

The information `ypbind` remembers is the association of a domain name with the Internet address of the NIS server, and the port on that host at which the `ypserv` process is listening for service requests. The process that `ypbind` goes through to find and remember this information is called *binding*, and this process is driven by client requests. As a request for an unbound domain arrives, the `ypbind` process broadcasts on the net to find a `ypserv` process that serves maps within that domain. Because the binding is established by broadcasting, at least one `ypserv` process must be on every net. After a domain is bound by a particular `ypbind`, that same binding is given to every client process on the node. The `ypbind` process on the local node or a remote node may be queried for the binding of a particular domain by using the `ypwhich`(1) command.

Bindings are verified before they are distributed to a client process. If `ypbind` cannot speak to the `ypserv` process to which it is bound, it marks the domain as unbound, tells the client process that the domain is unbound, and tries to rebind the domain again. Requests received for an unbound domain fail immediately. Generally, a bound domain is marked as unbound when the node running `ypserv` crashes or is overloaded. In this case, `ypbind` binds to any NIS server available on the net (typically, one that is not as heavily loaded).

The `ypbind` command also accepts requests to set its binding for a particular domain. Usually, the request is generated by the NIS subsystem. The `ypset`(8) command is a command to access the `Set_domain` facility. It is only for troubleshooting.

The `ypbind` and `ypserv` programs catch the `SIGHUP` signal and reregister themselves with `portmap` when they receive the signal. This enables these commands to continue running properly when `portmap` must be restarted.

FILES

`/etc/yp/ypserv.log`	The `ypserv` program appends its log messages to this file; however, you must first create this file with write access for `ypserv`.

SEE ALSO

`portmap`(8), `yppush`(8), `ypset`(8), `ypxfr`(8)

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

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

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

NAME

`ypset` – Points `ypbind` at a particular server

SYNOPSIS

`/etc/yp/ypset` [`-V1`] [`-h` *host*] [`-d` *domain*] *server*
`/etc/yp/ypset` [`-V2`] [`-h` *host*] [`-d` *domain*] *server*

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `ypset` command tells `ypbind` (see `ypserv`(8)) to get the network information service (NIS) for the specified *domain* from the `ypserv`(8) process running on *server*. If *server* is down, or is not running `ypserv`(8), this is not discovered until an NIS client process tries to get a binding for the domain. At this point, `ypbind` tests the binding set by `ypset`. If the binding is not valid, `ypbind` tries to rebind for the same domain.

The `ypset` command is useful for binding a client node that is not on a broadcast net, or that is on a broadcast net not running an NIS server host. It also is useful for debugging NIS client applications (for instance, where an NIS map exists only one NIS server host).

If you use ATM as your network interface, certain applications or daemons may not work as you expect if broadcast is necessary for that application or daemon. ATM is a point-to-point connection, and broadcast is not supported on ATM at this time.

When several hosts on the local net are supplying NIS services, `ypbind` can rebind to another host even as you are attempting to discover if the `ypset` operation succeeded. That is, you can type the following command lines:

```
ypset host1
ypwhich
```

The response is as follows:

```
host2
```

This response can be confusing. It is a function of the NIS subsystem's attempt to load balance among the available NIS servers, and it occurs when `host1` does not respond to `ypbind` because it is not running `ypserv`(8) (or is overloaded), and `host2`, running `ypserv`(8), gets the binding.

The *server* argument indicates the NIS server to which to bind, and you can specify it as a name or an Internet Protocol (IP) address. If specified as a name, `ypset` attempts to use NIS services to resolve the name to an IP address. This works only if the node has a current valid binding for the domain in question. Usually, you should specify *server* as an IP address.

See the `ypfiles`(5) and `ypserv`(8) man pages for an overview of the network information service (NIS).

The `ypset` command accepts the following options:

`-V1`	Binds *server* for the V.1 YP protocol.
`-V2`	Binds *server* for the V.2 YP protocol. If no version is supplied, `ypset` first attempts to set the domain for the V.2 protocol. If this attempt fails, `ypset` attempts to set the domain for the V.1 protocol.
`-h` *host*	Sets `ypbind` binding on *host* instead of locally. You can specify the *host* argument as a name or as an IP address.
`-d` *domain*	Specifies *domain* instead of the default domain.
server	Server name.

SEE ALSO

`ypserv`(8)

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

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

NAME

`ypstart` – Starts the network information service (NIS)

SYNOPSIS

`/etc/ypstart`

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `ypstart` script starts the software necessary for the network information service (NIS) at system startup when executed by the `netstart`(8) script.

The `ypstart` script performs the following functions:

- Sets the NIS domain name to the name found in the `/etc/config/ypdomain.txt` file (see `domainname`(1)).
- Starts the NIS server process (`ypserv`(8)).
- Binds the BNIS service to the local host (`ypbind` (see `ypserv`(8)) and verifies the binding (`ypwhich`(1)).
- Updates the `ypservers`, `passwd.byname`, and `passwd.byuid` domains (`ypxfr`(8)).

FILES

`/etc/config/ypdomain.txt`	Holds the NIS domain name

SEE ALSO

`netstart`(8), `sdaemon`(8), `ypserv`(8), `ypxfr`(8)

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

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

NAME

`ypupdated` – Updates NIS information

SYNOPSIS

`/etc/ypupdated` [`-i`] [`-s`]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `ypupdated` command is an Remote Procedure Call (RPC)-based daemon that updates the network information service (NIS) (formerly called yellow pages (YP)), and it is started up by `inetd`(8). `ypupdated` is always registered as RPC program 100028. `ypupdated` consults the `updaters`(5) file in the `/etc/yp` directory to determine which NIS maps should be updated and how to change them.

By default, the daemon requires the most secure method of authentication available to it, either Data Encryption Standard (DES) (secure) or UNIX (insecure).

The `ypupdated` command accepts the following options:

`-i` Accepts also RPC calls that have the insecure `AUTH_UNIX` credentials. This allows programmatic updating of NIS maps in all networks.

`-s` Accepts only calls authenticated by using the secure RPC mechanism (`AUTH_DES` authentication). This disables programmatic updating of NIS maps unless the network supports these calls.

FILES

`/etc/yp/updaters`	Map file

SEE ALSO

`inetd`(8), `keyserv`(8)

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

NAME

`ypxfr`, `ypxfr1pdy`, `ypxfr1phr`, `ypxfr2pdy` – Transfers network information service (NIS) map from NIS server

SYNOPSIS

`/etc/yp/ypxfr` [`-f`] [`-h` *host*] [`-d` *domain*] [`-c`] [`-C` *tid prog ipadd port*] *mapname*
`/etc/yp/ypxfr1pdy`
`/etc/yp/ypxfr1phr`
`/etc/yp/ypxfr2pdy`

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `ypxfr` command moves a network information service (NIS) map to the local host by making use of normal NIS services. It creates a temporary map in the `/etc/yp/domain` directory (which must already exist), fills it by enumerating the map's entries, fetches the map parameters (master and order number), and loads them. It then deletes any old versions of the map and moves the temporary map to the real *mapname*.

If `ypxfr` is run interactively, it writes its output to the terminal. However, if it is invoked without a controlling terminal, and if you have created log file `/etc/yp/ypxfr.log`, it appends all of its output to that file. Because `ypxfr` is most often run from `/usr/lib/crontab`, or by `ypserv`, you can use the log file to retain a record of what was attempted, and what the results were.

For consistency between servers, you should run `ypxfr` periodically for every map in the NIS database. Different maps change at different rates. For example, the `services.byname` map may not change for months at a time; therefore, it may be checked only once a day in the early morning hours. The `mail.aliases` or `hosts.byname` file changes several times per day. In such a case, you may want to check hourly for updates.

You can use a `crontab`(1) file entry to perform periodic updates automatically. However, rather than having a separate crontab entry for each map, you can group commands in a shell script to update several maps. The `/etc/yp/ypxfr1pdy`, `/etc/yp/ypxfr1phr`, and `/etc/yp/ypxfr2pdy` scripts allow you to run one transfer per day, to transfer volatile maps hourly, and to run two transfers per day, respectively. You can modify these scripts to meet your site's requirements.

See the `ypfiles`(5) and `ypserv`(8) man pages for an overview of the network information service.

The `ypxfr` command accepts the following options:

`-f` Forces the transfer to occur even if the version at the master is not more recent than the local version.

`-h` *host*	Gets the map from *host*, regardless of what the map says is the master. If you omit *host*, `ypxfr` asks the NIS service for the name of the master, and it tries to get the map from there. The *host* argument can be a name or an Internet address in the form *a.b.c.d*.
`-d` *domain*	Specifies a domain other than the default domain.
`-c`	Inhibits `Clear current map` request to the local `ypserv` process. Use this option if `ypserv` is not running locally at the time you are running `ypxfr`; otherwise, `ypxfr` complains that it cannot talk to the local `ypserv`, and the transfer fails.
`-C` *tid prog ipadd port*	For use by `ypserv`. When `ypserv` invokes `ypxfr`, it specifies that `ypxfr` should call back a `yppush`(8) process at the host that has Internet Protocol (IP) address *ipadd*, registered as program number *prog*, listening on port *port*, and waiting for a response to transaction *tid*.
mapname	Specifies name of map to be transferred.

FILES

`/etc/yp/ypxfr.log`	The `ypxfr` program appends its log messages to this file; however, you must first create this file with write access for `ypxfr`.
`/usr/lib/crontab`	File from which `ypxfr` is run.

SEE ALSO

`yppush`(8), `ypserv`(8)

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

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