IRIX System Administrator’s Reference Manual

Version 5.0

Document Number 007-0604-050

11/90
This document contains proprietary information of Silicon Graphics, Inc. The contents of this document may not be disclosed to third parties, copied or duplicated in any form, in whole or in part, without the prior written permission of Silicon Graphics, Inc.

Restricted Rights Legend

Use, duplication or disclosure of the technical data contained in this document by the Government is subject to restrictions as set forth in subdivision (c) (1) (ii) of the Rights in Technical Data and Computer Software clause at DFARS 52.227-7013, and/or in similar or successor clauses in the FAR, or the DOD or NASA FAR Supplement. Unpublished rights reserved under the Copyright Laws of the United States. Contractor/manufacturer is Silicon Graphics, Inc., 2011 N. Shoreline Blvd., Mountain View, CA 94039-7311.

IRIX System Administrator's Reference Manual
Version 5.0
Document Number 007-0604-050

Silicon Graphics, Inc.
Mountain View, California

IRIX is a trademark of Silicon Graphics, Inc.
UNIX is a trademark of AT&T, Inc.
TABLE OF CONTENTS

1M. System Administration

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>intro</td>
<td>introduction to commands and programs</td>
</tr>
<tr>
<td>accept</td>
<td>allow or prevent LP requests</td>
</tr>
<tr>
<td>acct</td>
<td>overview of miscellaneous accounting commands</td>
</tr>
<tr>
<td>acctcms</td>
<td>command summary from accounting records</td>
</tr>
<tr>
<td>acctcon</td>
<td>connect-time accounting</td>
</tr>
<tr>
<td>acctmerg</td>
<td>merge or add total accounting files</td>
</tr>
<tr>
<td>acctprc</td>
<td>process accounting</td>
</tr>
<tr>
<td>acctsh</td>
<td>shell procedures for accounting</td>
</tr>
<tr>
<td>addclient</td>
<td>allow remote printing clients to connect</td>
</tr>
<tr>
<td>arp</td>
<td>address resolution display and control</td>
</tr>
<tr>
<td>bootp</td>
<td>server for Internet Bootstrap Protocol</td>
</tr>
<tr>
<td>brc</td>
<td>system initialization procedures</td>
</tr>
<tr>
<td>captoinfo</td>
<td>convert a termcap description into a terminfo</td>
</tr>
<tr>
<td>chkconfig</td>
<td>configuration state checker</td>
</tr>
<tr>
<td>chroot</td>
<td>change root directory for a command</td>
</tr>
<tr>
<td>chrtbl</td>
<td>generate character classification tables</td>
</tr>
<tr>
<td>ckbupsd</td>
<td>check file system backup schedule</td>
</tr>
<tr>
<td>cli</td>
<td>clear i-node</td>
</tr>
<tr>
<td>cron</td>
<td>clock daemon</td>
</tr>
<tr>
<td>dd</td>
<td>convert and copy a file</td>
</tr>
<tr>
<td>devnm</td>
<td>device name</td>
</tr>
<tr>
<td>dglid</td>
<td>Distributed Graphics Library server</td>
</tr>
<tr>
<td>diskusg</td>
<td>generate disk accounting data by user ID</td>
</tr>
<tr>
<td>distcp</td>
<td>copy software distribution</td>
</tr>
<tr>
<td>du</td>
<td>summarize disk usage</td>
</tr>
<tr>
<td>dvhtool</td>
<td>modify and obtain disk volume header information</td>
</tr>
<tr>
<td>fingerd</td>
<td>remote user information server</td>
</tr>
<tr>
<td>fsck</td>
<td>check and repair file systems</td>
</tr>
<tr>
<td>fstat</td>
<td>report file system status</td>
</tr>
<tr>
<td>fstyp</td>
<td>determine file system identifier</td>
</tr>
<tr>
<td>ftpd</td>
<td>Internet File Transfer Protocol server</td>
</tr>
<tr>
<td>fuser</td>
<td>identify processes using a file or file structure</td>
</tr>
<tr>
<td>fwtmp</td>
<td>manipulate connect accounting records</td>
</tr>
<tr>
<td>fx</td>
<td>disk utility</td>
</tr>
<tr>
<td>gated</td>
<td>gateway routing daemon</td>
</tr>
<tr>
<td>getty</td>
<td>set terminal type, modes, speed, and line discipline</td>
</tr>
<tr>
<td>growfs</td>
<td>expand a filesystem</td>
</tr>
<tr>
<td>halt</td>
<td>halt the system</td>
</tr>
<tr>
<td>hinv</td>
<td>hardware inventory command</td>
</tr>
<tr>
<td>hyroute</td>
<td>set the HyperNet routing tables</td>
</tr>
<tr>
<td>ifconfig</td>
<td>configure network interface parameters</td>
</tr>
<tr>
<td>inetd</td>
<td>Internet &quot;super-server&quot;</td>
</tr>
</tbody>
</table>

April 1990

- i -

Version 5.0
Table of Contents

infocmp compare or print out terminfo descriptions
init process control initialization
inst software installation tool
killall kill named processes
labelit provide labels for file systems
lboot configure bootable kernel
link link and unlink files and directories
lpadmin configure the LP spooling system
lpc line printer control program
lpd line printer daemon
lpsched start/stop the LP scheduler and move requests
lvck check and restore consistency of logical volumes
lvinit initialize logical volume devices.
makedev Create device special files
makewhatis make manual page
mkboottape make a boot tape
mkcentpr register a color Centronics printer with LP
mkfile create a file
mkfs construct a file system
mklv construct or extend a logical volume
mknod build special file or named pipe (FIFO)
mkps register a LaserWriter printer with LP
mount mount, unmount multiple file systems
mountall mount and dismount filesystems
mrouted IP multicast routing daemon
multi switch the system to multi-user mode
mvdir move a directory
named Internet domain name server
ncheck generate path names from i-numbers
network network initialization and shutdown script
newaliases rebuild the data base for the mail aliases file
nvram get or set non-volatile RAM variable(s)
pac printer/plotter accounting information
passmgmt password file management
ping send ECHO_REQUEST packets to network hosts
portmap TCP,UDP port to RPC program number mapper
powerdown stop all processes and halt the system
preset reset the lp queue system to a pristine state
profiler UNIX system profiler
prtvtoc print volume header information.
pwck password/group file checkers
rc0 run commands performed to stop the operating system
rc2 run commands performed for multi-user environment
reboot reboot the system
renice ................................ alter priority of running processes
rexecd ................................ remote execution server
rlogind ................................ remote login server
rmail ................................ receive mail via UUCP
rmprinter ................................. remove a printer from the LP spooling system
rmt ....................................... remote magtape protocol module
route .................................... manually manipulate the routing tables
routed .................................... network routing daemon
rpcinfo ................................ report RPC information
rshd ..................................... remote shell server
runacct ................................ run daily accounting
rwhod ................................... system status server
sar ....................................... system activity report package
savecore ................................ save a core dump of the operating system
sendmail ................................. send mail over the internet
setmnt .................................. establish mount table
setsym .................................. set up a debug kernel for symbolic debugging
shutdown ............................... shut down system, change system state
single .................................. switch the system to single-user mode
su ....................................... become super-user or another user
swap .................................... swap administrative interface
sync ..................................... update the super block
syslogd ................................ log systems messages
tabled .................................. tablet reader daemon tablet/digitizers
talkd ................................... remote user communication server
telnetd ................................ Internet TELNET protocol server
tftpd ................................... Internet Trivial File Transfer Protocol server
tic ...................................... terminfo compiler
timed ................................... time server daemon
timedc .................................. timed control program
timeslave ............................... 'slave' local clock to a better one
uadmin .................................. administrative control
uucheck ................................ check the uucp directories and permissions file
uucico ................................ file transport program for the uucp system
uucleanup ............................... uucp spool directory clean-up
ugetty .................................. set terminal type, modes, speed, and line discipline
uusched ................................ the scheduler for the uucp file transport program
uutry .................................. try to contact remote system with debugging on
uuxqt .................................. execute remote command requests
versions ................................ software versions tool
whodo .................................. who is doing what
### 7m. Special Files

- **intro**: introduction to special files
- **arp**: Address Resolution Protocol
- **audio**: bi-directional audio channel interface
- **cdsio**: 6-port serial I/O
- **clone**: open any minor device on a STREAMS driver
- **console**: console interface
- **dks**: Small Computer Systems Interface (SCSI) disk driver
- **dn_ll**: 4DDN special files
- **drain**: capture unimplemented link-layer protocols
- **ds**: generic (user mode) SCSI driver
- **duart**: on-board serial ports
- **ethernet**: IRIS-4D Series ethernet controllers
- **gpib**: driver for NI VME IEEE-488 controller
- **gse**: Silicon Graphics 5080 workstation interface card
- **hl**: hardware spinlocks driver
- **hy**: HyperNet interface
- **icmp**: Internet Control Message Protocol
- **ik**: Ikon 10088 hardcopy interface controller
- **imon**: inode monitor device
- **inet**: Internet protocol family
- **ip**: Internet Protocol
- **ipi**: Xylogics IPI disk controllers and driver.
- **ips**: Interphase disk controllers and driver.
- **keyboard**: keyboard specifications
- **klog**: kernel error logging interface
- **lv**: logical volume Disk driver
- **mem**: core memory
- **mouse**: optical mouse specifications
- **mtio**: magnetic tape interface
- **netintro**: introduction to networking facilities
- **null**: the null file
- **plp**: parallel line printer interface
- **prf**: operating system profiler
- **pty**: pseudo terminal driver
- **raw**: raw network protocol family
- **root**: Partition names.
- **sa**: devices administered by System Administration
- **smfd**: SCSI floppy disk driver
- **snoop**: network monitoring protocol
- **streamio**: STREAMS ioctl commands
- **t3270**: Silicon Graphics 3270 interface card
- **tcp**: Internet Transmission Control Protocol
- **termio**: general System V and POSIX terminal interfaces
- **tps**: SCSI 1/4-inch Cartridge tape interface
ts . . . . ISI VME-QIC2/X cartridge tape controller
tty . . . . . . . . . . . . controlling terminal interface
udp . . . . . . . . . . . . Internet User Datagram Protocol
vh . . . . . . . . . . . . . . . . . . . . disk volume header.
xmt . . . . Xylogics 1/2 inch magnetic tape controller
xyl . . . . . . . . Xylogics disk controllers and driver.
zero . . . . . . . . . . . . . . . . . . . . source of zeroes
NAME
intro – introduction to maintenance commands and application programs

DESCRIPTION
This section describes, in alphabetical order, commands that are used chiefly for system maintenance and administration purposes. The commands in this section should be used along with those listed in Section 1 of the User's Reference Manual and Sections 1, 2, 3, 4, and 5 of the Programmer's Reference Manual. References of the form name(1), (2), (3), (4) and (5) refer to entries in the above manuals. References of the form name(1M), name(7) or name(8) refer to entries in this manual.

COMMAND SYNTAX
Unless otherwise noted, commands described in this section accept options and other arguments according to the following syntax:

name [option(s)] [cmdarg(s)]

where:

name The name of an executable file.

option – noargletter(s) or,
– argletter<>optarg
where <> is optional white space.

noargletter A single letter representing an option without an argument.

argletter A single letter representing an option requiring an argument.

optarg Argument (character string) satisfying preceding argletter.

cmdarg Path name (or other command argument) not beginning with – or, – by itself indicating the standard input.

SEE ALSO
getopt(3C) in the Programmer's Reference Manual.

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

Regrettably, not all commands adhere to the aforementioned syntax.
NAME
accept, reject – allow or prevent LP requests

SYNOPSIS
/usr/lib/accept destinations
/usr/lib/reject [-r[ reason ]] destinations

DESCRIPTION
accept allows lp(1) to accept requests for the named destinations. A destination can be either a line printer (LP) or a class of printers. Use lpstat(1) to find the status of destinations.

Reject prevents lp(1) from accepting requests for the named destinations. A destination can be either a printer or a class of printers. Use lpstat(1) to find the status of destinations. The following option is useful with reject.

-rr[ reason ] Associates a reason with preventing lp from accepting requests. This reason applies to all printers mentioned up to the next -r option. Reason is reported by lp when users direct requests to the named destinations and by lpstat(1). If the -r option is not present or the -r option is given without a reason, then a default reason will be used.

FILES
/usr/spool/lp/*

SEE ALSO
lpadmin(1M), lpsched(1M).
NAME
acctdisk, acctdusg, accton, acctwtmp — overview of accounting and miscellaneous accounting commands

SYNOPSIS
/usr/lib/acct/acctdisk
/usr/lib/acct/acctdusg [−u file] [−p file]
/usr/lib/acct/accton [file]
/usr/lib/acct/acctwtmp "reason"

DESCRIPTION
Accounting software is structured as a set of tools (consisting of both C programs and shell procedures) that can be used to build accounting systems. acctsh(1M) describes the set of shell procedures built on top of the C programs.

Connect time accounting is handled by various programs that write records into /etc/utmp, as described in utmp(4). The programs described in acctcon(1M) convert this file into session and charging records, which are then summarized by acctmerg(1M).

Process accounting is performed by the UNIX system kernel. Upon termination of a process, one record per process is written to a file (normally /usr/adm/pacct). The programs in acctproc(1M) summarize this data for charging purposes; acctcms(1M) is used to summarize command usage. Current process data may be examined using acctcom(1).

Process accounting and connect time accounting (or any accounting records in the format described in acct(4)) can be merged and summarized into total accounting records by acctmerg (see tacct format in acct(4)). prtacct (see acctsh(1M)) is used to format any or all accounting records.

acctdisk reads lines that contain user ID, login name, and number of disk blocks and converts them to total accounting records that can be merged with other accounting records.

acctdusg reads its standard input (usually from find /−print) and computes disk resource consumption (including indirect blocks) by login. If −u is given, records consisting of those filenames for which acctdusg charges no one are placed in file (a potential source for finding users trying to avoid disk charges). If −p is given, file is the name of the password file. This option is not needed if the password file is /etc/passwd. (See diskusg(1M) for more details.)
accton alone turns process accounting off. If file is given, it must be the name of an existing file, to which the kernel appends process accounting records (see acct(2) and acct(4)).

acctwtmp writes a utmp(4) record to its standard output. The record contains the current time and a string of characters that describe the reason. A record type of ACCOUNTING is assigned (see utmp(4)). reason must be a string of 11 or fewer characters, numbers, $, or spaces. The accounting startup and shutdown scripts /usr/lib/acct/startup and /usr/lib/acct/shutacct use the acctwtmp command to record system startup and shutdown events.

NOTE

The file /etc/config/acct controls the automatic startup and periodic report generation of the accounting subsystem. If this file contains the flag value "on", process accounting will be enabled by /etc/init.d/acct each time the system is brought up, and nightly reports will be generated and placed in the directory /usr/adm/acct.

/etc/chkconfig should be used to modify the contents of the /etc/config/acct file.

FILES

/etc/passwd used for login name to user ID conversions
/usr/lib/acct holds all accounting commands listed in
    sub-class 1M of this manual
/usr/adm/pacct current process accounting file
/etc/wtmp login/logoff history file
/etc/config/acct if contains "on", accounting runs automatically

SEE ALSO

acctcms(1M), acctcon(1M), acctmerg(1M), acctprc(1M), acctsh(1M),
    diskusg(1M), fwtmp(1M), runacct(1M), chkconfig(1M)
acctcom(1) in the User's Reference Manual
NAME
acctcms – command summary from per-process accounting records

SYNOPSIS
/usr/lib/acct/acctcms [options] files

DESCRIPTION
acctcms reads one or more files, normally in the form described in acct(4). It adds all records for processes that executed identically-named commands, sorts them, and writes them to the standard output, normally using an internal summary format. The options are:

-a Print output in ASCII rather than in the internal summary format. The output includes command name, number of times executed, total kcore-minutes, total CPU minutes, total real minutes, mean size (in K), mean CPU minutes per invocation, "hog factor", characters transferred, and blocks read and written, as in acctcom(1). Output is normally sorted by total kcore-minutes.
-c Sort by total CPU time, rather than total kcore-minutes.
-j Combine all commands invoked only once under "***other".
-n Sort by number of command invocations.
-s Any filenames encountered hereafter are already in internal summary format.
-t Process all records as total accounting records. The default internal summary format splits each field into prime and non-prime time parts. This option combines the prime and non-prime time parts into a single field that is the total of both, and provides upward compatibility with old (i.e., UNIX System V) style acctcms internal summary format records.

The following options may be used only with the -a option.

-p Output a prime-time-only command summary.
-o Output a non-prime (offshift) time only command summary.

When -p and -o are used together, a combination prime and non-prime time report is produced. All the output summaries will be total usage except number of times executed, CPU minutes, and real minutes, which will be split into prime and non-prime.

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

acctcms file ... > today
cp total previous total
acctcms -s today previous total > total
acctcms -a -s today
SEE ALSO
acct(1M), acctcon(1M), acctmerg(1M), acctprc(1M), acctsh(1M),
fwtmp(1M), runacct(1M)
acctcom(1) in the User's Reference Manual

BUGS
Unpredictable output results if -t is used on new style internal summary format files, or if it is not used with old style internal summary format files.
NAME
acctcon1, acctcon2 – connect-time accounting

SYNOPSIS
/usr/lib/acct/acctcon1 [options]
/usr/lib/acct/acctcon2

DESCRIPTION
acctcon1 converts a sequence of login/logoff records read from its standard input to a sequence of records, one per login session. Its input should normally be redirected from /etc/wtmp. Its output is ASCII, giving device, user ID, login name, prime connect time (seconds), non-prime connect time (seconds), session starting time (numeric), and starting date and time. The options are:

-p  Print input only, showing line name, login name, and time (in both numeric and date/time formats).
-t  acctcon1 maintains a list of lines on which users are logged in. When it reaches the end of its input, it emits a session record for each line that still appears to be active. It normally assumes that its input is a current file, so that it uses the current time as the ending time for each session still in progress. The -t flag causes it to use, instead, the last time found in its input, thus assuring reasonable and repeatable numbers for non-current files.

-l file  file is created to contain a summary of line usage showing line name, number of minutes used, percentage of total elapsed time used, number of sessions charged, number of logins, and number of logoffs. This file helps track line usage, identify bad lines, and find software and hardware oddities. Hangup, termination of login(1) and termination of the login shell each generate logoff records, so that the number of logoffs is often three to four times the number of sessions. See init(1M) and utmp(4).

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

acctcon2 expects as input a sequence of login session records and converts them into total accounting records (see tacct format in acct(4)).

EXAMPLES
These commands are typically used as shown below. The file ctmp is created only for the use of acctprc(1M) commands:

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

SEE ALSO
acct(1M), acctcms(1M), acctmerg(1M), acctproc(1M), acctsh(1M), fwtmp(1M), init(1M), runacct(1M)
acctcom(1), login(1) in the User’s Reference Manual

BUGS
The line usage report is confused by date changes. Use wtmpfix (see fwtmp(1M)) to correct this situation.
NAME
acctmerg – merge or add total accounting files

SYNOPSIS
/usr/lib/acct/acctmerg [options] [file] . . .

DESCRIPTION
acctmerg reads its standard input and up to nine additional files, all in the
tacct format (see acct(4)) or an ASCII version thereof. It merges these
inputs by adding records whose keys (normally user ID and name) are
identical, and expects the inputs to be sorted on those keys. Options are:

-a Produce output in ASCII version of tacct.
-i Input files are in ASCII version of tacct.
-p Print input with no processing.
-t Produce a single record that totals all input.
-u Summarize by user ID, rather than user ID and name.
-v Produce output in verbose ASCII format, with more precise notation
for floating-point numbers.

EXAMPLES
The following sequence is useful for making "repairs" to any file kept in
this format:

acctmerg -v <file1 > file2

Edit file2 as desired . . .

acctmerg -i <file2 > file1

SEE ALSO
acct(1M), acctcms(1M), acctcon(1M), acctprc(1M), acctsh(1M),
fwtmp(1M), runacct(1M)
acctcom(1) in the User's Reference Manual
NAME
acctprc1, acctprc2 – process accounting

SYNOPSIS
/usr/lib/acct/acctprc1 [ctmp]
/usr/lib/acct/acctprc2

DESCRIPTION
acctprc1 reads input in the form described by acct(4), adds login names corresponding to user IDs, then writes for each process an ASCII line giving user ID, login name, prime CPU time (tics), non-prime CPU time (tics), and mean memory size (in memory segment units). If ctmp is given, it is expected to contain a list of login sessions, in the form described in acctcon(1M), sorted by user ID and login name. If this file is not supplied, it obtains login names from the password file. The information in ctmp helps it distinguish among different login names that share the same user ID.

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

These commands are typically used as shown below:

acctprc1 ctmp </usr/adm/pacct | acctprc2 >ptacct

FILES
/etc/passwd

SEE ALSO
acct(1M), acctcms(1M), acctcon(1M), acctmerg(1M), acctsh(1M), cron(1M), fwtmp(1M), runacct(1M)
acctcom(1) in the User’s Reference Manual

BUGS
Although it is possible to distinguish among login names that share user IDs for commands run normally, it is difficult to do this for those commands run from cron(1M), for example. More precise conversion can be done by faking login sessions on the console via the acctwtmp program in acct(1M).

CAVEAT
A memory segment of the mean memory size is a unit of measure for the number of bytes in a logical memory segment on a particular processor.
NAME

chargefee, ckpacct, dodisk, lastlogin, monacct, nulladm, prctmp, prdaily,
prtacct, runacct, shutacct, startup, turnacct – shell procedures for account-
ing

SYNOPSIS

/usr/lib/acct/chargefee login-name number
/usr/lib/acct/ckpacct [blocks]
/usr/lib/acct/dodisk [-o] [files ...]
/usr/lib/acct/lastlogin
/usr/lib/acct/monacct number
/usr/lib/acct/nulladm file
/usr/lib/acct/prctmp
/usr/lib/acct/prdaily [-l] [-c] [mmdd ]
/usr/lib/acct/prtacct file [ "heading"
/usr/lib/acct/runacct [mmdd] [mmdd state]
/usr/lib/acct/shutacct [ "reason"
/usr/lib/acct/startup
/usr/lib/acct/turnacct on | off | switch

DESCRIPTION

chargefee can be invoked to charge a number of units to login-name. A
record is written to /usr/adm/fee, to be merged with other accounting
records during the night.

ckpacct should be initiated via cron(1M). It periodically checks the size of
/usr/adm/pacct. If the size exceeds blocks, 1000 by default, turnacct will
be invoked with argument switch. If the number of free disk blocks in the
/usr file system falls below 500, ckpacct will automatically turn off the col-
clection of process accounting records via the off argument to turnacct.
When at least this number of blocks is restored, the accounting will be
activated again. This feature is sensitive to the frequency at which ckpacct
is executed, usually by cron.

dodisk should be invoked by cron to perform the disk accounting functions.
By default, it will do disk accounting on the special files in /etc/fstab. If the
-o flag is used, it will do a slower version of disk accounting by login direc-
tory. files specifies the one or more filesystem names where disk account-
ing will be done. If files are used, disk accounting will be done on these
filesystems only. If the -o flag is used, files should be mount points of
mounted filesystems. If omitted, they should be the special file names of
mountable filesystems.

`lastlogin` is invoked by `runacct` to update `/usr/adm/acct/sum/loginlog`, which shows the last date on which each person logged in.

`monacct` should be invoked once each month or each accounting period. `number` indicates which month or period it is. If `number` is not given, it defaults to the current month (01–12). This default is useful if `monacct` is to executed via `cron(1M)` on the first day of each month. `monacct` creates summary files in `/usr/adm/acct/fiscal` and restarts summary files in `/usr/adm/acct/sum`.

`nulladm` creates file with mode 664 and ensures that owner and group are `adm`. It is called by various accounting shell procedures.

`prctmp` can be used to print the session record file (normally `/usr/adm/acct/nite/ctmp` created by `acctcon(1M)`).

`prdaily` is invoked by `runacct` to format a report of the previous day’s accounting data. The report resides in `/usr/adm/acct/sum/rprt mndd` where `mndd` is the month and day of the report. The current daily accounting reports may be printed by typing `prdaily`. Previous days’ accounting reports can be printed by using the `mndd` option and specifying the exact report date desired. The `-I` flag prints a report of exceptional usage by login id for the specified date. Previous daily reports are cleaned up and therefore inaccessible after each invocation of `monacct`. The `-c` flag prints a report of exceptional resource usage by command, and may be used on current day’s accounting data only.

`prtacct` can be used to format and print any total accounting (`tacct`) file.

`runacct` performs the accumulation of connect, process, fee, and disk accounting on a daily basis. It also creates summaries of command usage. For more information, see `runacct(1M)`.

`shutacct` is invoked during a system shutdown to turn process accounting off and append a "reason" record to `/etc/utmp`.

`startup` can be called to turn the accounting on when the system is brought to a multi-user state.

`turnacct` is an interface to `accton` (see `acct(1M)`) to turn process accounting on or off. The `switch` argument turns accounting off, moves the current `/usr/adm/pacct` to the next free name in `/usr/adm/pacctincr` (where `incr` is a number starting with `I` and incrementing by one for each additional `pacct` file), then turns accounting back on again. This procedure is called by `ckpacct` and thus can be taken care of by the `cron` and used to keep `pacct` to a reasonable size. `acct` starts and stops process accounting via `init` and `shutdown` accordingly.
FILES

/usr/adm/acctime accumulator for fees
/usr/adm/pacct current file for per-process accounting
/usr/adm/pacct* used if pacct gets large and during
execution of daily accounting procedure
/etc/wtmp login/logoff summary
/usr/lib/acct/ptelus.awk contains the limits for exceptional
usage by login id
/usr/lib/acct/ptecms.awk contains the limits for exceptional
usage by command name
/usr/adm/acct/nite working directory
/usr/lib/acct holds all accounting commands listed in
/sub-class 1M of this manual
/usr/adm/acct/sum summary directory, should be saved

SEE ALSO

acct(1M), acctems(1M), acctcon(1M), acctmerg(1M), acctprc(1M),
cron(1M), diskusg(1M), fwtmp(1M), runacct(1M)
acctcom(1) in the User’s Reference Manual
NAME
addclient – allow remote printing clients to connect

SYNOPSIS
addclient client
addclient -a

DESCRIPTION
This command is used on an IRIS that has a printer attached to it. The command registers a remote user’s machine, allowing it to connect to this machine in order to submit a printing request using \textit{lp}(1). In the first form of the command, \textit{client} is the name of a remote machine to which access will be granted. If the second form of the command is used then all remote machines will be granted access. This command modifies the \textit{.rhosts} file in the home directory of user \textit{lp} and possibly \textit{/etc/passwd}.

Execute this command once for each client machine that will use the network to submit printing jobs on this machine, or use the \texttt{-a} option to allow universal access.

FILES
/etc/passwd
/usr/spool/lp/.rhosts

SEE ALSO
mknetpr(1M), mkcentpr(1M), mkPS(1M).
NAME

arp – address resolution display and control

SYNOPSIS

arp hostname
arp -a [ unix ] [ kmem ]
arp -d hostname
arp -s hostname ether_addr [ temp ] [ pub ] [ trail ]
arp -f filename

DESCRIPTION

The *arp* program displays and modifies the Internet-to-Ethernet address translation tables used by the address resolution protocol (*arp(7P)*).

With no flags, the program displays the current ARP entry for *hostname*. The host may be specified by name or by number, using Internet dot notation. With the -a flag, the program displays all of the current ARP entries by reading the table from the file *kmem* (default /dev/kmem) based on the kernel file *unix* (default /unix).

With the -d flag, a super-user may delete an entry for the host called *hostname*.

The -s flag is given to create an ARP entry for the host called *hostname* with the Ethernet address *ether_addr*. The Ethernet address is given as six hex bytes separated by colons. The entry will be permanent unless the word *temp* is given in the command. If the word *pub* is given, the entry will be "published"; i.e., this system will act as an ARP server, responding to requests for *hostname* even though the host address is not its own. The word *trail* indicates that trailer encapsulations may be sent to this host.

The -f flag causes the file *filename* to be read and multiple entries to be set in the ARP tables. Entries in the file should be of the form

```
hostname ether_addr [ temp ] [ pub ] [ trail ]
```

with argument meanings as given above.

SEE ALSO

inet(7P), arp(7P), ifconfig(1M)
NAME

`bootp` – server for Internet Bootstrap Protocol

SYNOPSIS

```
/usr/etc/bootp [ -d ] [ -f ]
```

DESCRIPTION

`Bootp` is a server which supports the Internet Bootstrap Protocol (BOOTP). This protocol is designed to allow a (possibly diskless) client machine to determine its own Internet address, the address of a boot server and the name of an appropriate boot file to be loaded and executed. BOOTP does not provide the actual transfer of the boot file, which is typically done with a simple file transfer protocol such as TFTP. A detailed protocol specification for BOOTP is contained in RFC 951, which available from the Network Information Center.

The BOOTP protocol uses UDP/IP as its transport mechanism. The BOOTP server receives service requests at the UDP port indicated in the "bootp" service description contained in the file `/etc/services` (see `services(4)`). The BOOTP server is started by `inetd(1M)`, as configured in the `inetd.conf` file.

The basic operation of the BOOTP protocol is a single packet exchange as follows:

1) The booting client machine broadcasts a BOOTP request packet to the BOOTP server UDP port, using a UDP broadcast or the equivalent thereof. The request packet includes the following information:

   requester’s Ethernet address
   requester’s Internet address (optional)
   desired server’s name (optional)
   boot file name (optional)

2) All the BOOTP servers on the same Ethernet wire as the client machine receive the client’s request. If the client has specified a particular server, then only that server will respond.

3) The server looks up the requester in its configuration file by Internet address or Ethernet address, in that order of preference. (The BOOTP configuration file is described below.) If the Internet address was not specified by the requester and a configuration record is not found, the server will look in the `/etc/ethers` file (see `ethers(4)`) for an entry with the client’s Ethernet address. If an entry is found, the server will check the hostname of that entry against the `/etc/hosts` file (see `hosts(4)`) in order to complete the Ethernet address to Internet address mapping. If the BOOTP
request does not include the client’s Internet address and the server
is unable to translate the client’s Ethernet address into an Internet
address by either of the two methods described, the server will not
respond to the request.

4) The server performs name translation on the boot filename
requested and then checks for the presence of that file. If the file is
present, then the server will send a response packet to the requester
which includes the following information:

the requester’s Internet address
the server’s Internet address
the Internet address of a gateway to the server
the server’s name
vendor specific information (not defined by the protocol)

If the boot file is missing, the server will return a response packet
with a null filename, but only if the request was specifically
directed to that server. The pathname translation is: if the boot
filename is rooted, use it as is; else concatenate the root of the boot
subtree, as specified by the BOOTP configuration file, followed by
the filename supplied by the requester, followed by a period and
the requester’s hostname. If that file is not present, remove the
trailing period and host name and try again. If no boot filename is
requested, use the default boot file for that host from the
configuration table. If there is no default specified for that host,
use the general default boot filename, first with .hostname as a
suffix and then without.

Options
The -d option causes bootp to generate debugging messages. All messages
from bootp go through syslogd(1M), the system logging daemon.

The -f option enables the forwarding function of bootp. Refer to the fol-
lowing section on Booting Through Gateways for an explanation.

Bootp Configuration File
In order to perform its name translation and address resolution functions,
bootp requires configuration information, which it gets from an ASCII file
called /usr/etc/bootptab and from other system configuration files like
/etc/ethers and /etc/hosts. Here is a sample bootptab file:

```
# /usr/etc/bootptab: database for bootp server
#
# Blank lines and lines beginning with '#’ are ignored.
#
```

April 1990 - 2 - Version 5.0
# root of boot subtree

/users/local/boot

# default bootfile

unix

%%% 

# The remainder of this file contains one line per client interface
# with the information shown by the table headings below.
# The 'host' name is also tried as a suffix for the 'bootfile'
# when searching the boot directory. (e.g., bootfile.host)
#
# host  htype  haddr  iaddr  bootfile
#
unixbox 1 8:2:3:4:bb:cc 89.0.0.2

The fields of each line may be separated by variable amounts of white space (blanks and tabs). The first section, up to the line beginning '%%%', defines the place where bootp looks for boot files when the client requests a boot file using a non-rooted pathname. The second section of the file is used for mapping client Ethernet addresses into Internet addresses. The htype field should always have a value of 1 for now, which indicates that the hardware address is a 48-bit Ethernet address. The haddr field is the Ethernet address of the system in question expressed as 6 hex bytes separated by colons. The iaddr field is the 32-bit Internet address of the system expressed in standard dot notation (4 byte values in decimal, in network order, separated by periods). Each line in the second section can also specify a default boot file for each specific host. In the example above, if the host called unixbox makes a BOOTP request with no boot file specified, the server will select the first of the following that it finds:

/users/local/boot/unix.unixbox
/users/local/boot/unix

It is not necessary to create a record for every potential client in the bootptab file. The only constraint is that bootp will only respond to a request from a client if it can deduce the client's Internet address. There are three ways that this can happen: 1) the client already knows his Internet address and includes it in the BOOTP request packet, 2) there is an entry in /usr/etc/bootptab that matches the client's Ethernet address or 3) there are
entries in the /etc/ethers and /etc/hosts files (or their Yellow Pages equivalents) that allow the client's Ethernet address to be translated into an Internet address.

Booting Through Gateways
Since the BOOTP request is distributed using a UDP broadcast, it will only be received by other hosts on the same Ethernet cable as the client. In some cases the client may wish to boot from a host on another network. This can be accomplished by using the forwarding function of BOOTP servers on the local wire. To use BOOTP forwarding, there must be a bootp process running in a gateway machine on the local cable. A gateway machine is simply a machine with more than one Ethernet controller board. The gateway bootp must be invoked with the -f option to activate forwarding. Such a forwarding bootp will resend any BOOTP request it receives that asks for a specific host by name, if that host is on a different network from the client that sent the request. The BOOTP server forwards the packet using the full routing capabilities of the underlying IP layer in the kernel, so the forwarded packet will automatically be routed to the requested BOOTP server provided that the kernel routing tables contain a route to the destination network.

DIAGNOSTICS
The BOOTP server sends any messages it wants to reach the outside world through the system logging daemon, syslogd(1M). The actual disposition of these messages depends on the configuration of syslogd on the machine in question. Consult syslogd(1M) for further information.

Bootp can produce the following messages:

'get interface config' ioctl failed (message)
'get interface netmask' ioctl failed (message)
getsockname fails (message)
forwarding failed (message)
send failed (message)
set arp ioctl failed

Each of the above messages mean that a system call has returned an error unexpectedly. Such errors usually cause bootp to terminate. The message will be the result of calling perror(3) with the errno value that was returned.

less than two interfaces, -f flag ignored
Warning only. Means that the -f option was specified on a machine that is not a gateway. Forwarding only works on gateways.
request for unknown host xxx from yyy
    Information only. A BOOTP request was received asking for host xxx, but that host is not in the host database. The request was generated by yyy, which may be given as a host name or an Internet address.

request from xxx for 'fff'
    Information only. Bootp logs each request for a boot file. The means that host xxx has requested boot file fff.

boot file fff missing
    A request has been received for the boot file fff, but that file doesn't exist.

reply file fff
    Information only. Bootp has selected the file fff as the boot file to satisfy a request.

forward request with gateway address already set (dd.dd.dd.dd)
    The server has received a reply to be forwarded to a requester, but some other bootp has already filled himself in as the gateway. This is an error in the BOOTP forwarding mechanism.

missing gateway address
    This means that this bootp has generated a response to a client and is trying to send the response directly to the client (i.e. the request did not get forwarded by another bootp), but none of the Ethernet interfaces on this machine is on the same wire as the client machine. This indicates a bug in the BOOTP forwarding mechanism.

can't open /usr/etc/boottab
    The bootp configuration file is missing or has wrong permissions.

(re)reading /usr/etc/boottab
    Information only. Bootp checks the modification date of the configuration file on the receipt of each request and rereads it if it has been modified since the last time it was read.

bad hex address: xxx at line nnn of boottab
bad internet address: sss at line nnn of boottab
string truncated: sss, on line nnn of boottab
    These messages all mean that the format of the BOOTP configuration file is not valid.

'hosts' table length exceeded
    There are too many lines in the second section of the BOOTP configuration file. The current limit is 512.
can’t allocate memory
A call to malloc(3) failed.

gethostbyname(sss) fails (message)
A call to gethostbyname(3N) with the argument sss has failed.

gethostbyaddr(dd.dd.dd.dd) fails (message)
A call to gethostbyaddr(3N) with the argument dd.dd.dd.dd has failed.

can’t find source net for address xxx
This means that the server has received a datagram with a source address that doesn’t make sense. The offending address is printed as a 32 bit hexadecimal number xxx.

SEE ALSO
inetd(1M), syslogd(1M), tftp(1M), ethers(4), hosts(4), services(4)
NAME
  brc, bcheckrc – system initialization procedures

SYNOPSIS
  /etc/brc
  /etc/bcheckrc

DESCRIPTION
  These shell procedures are executed via entries in /etc/inittab by init(1M)
  whenever the system is booted (or rebooted).

  First, the bcheckrc procedure checks the status of the root file system. If the
  root file system is found to be bad, bcheckrc repairs it.

  Then, the brc procedure clears the mounted file system table, /etc/mtab and
  puts the entry for the root file system into the mount table.

  After these two procedures have executed, init checks for the initdefault
  value in /etc/inittab. This tells init in which run level to place the system.
  Since initdefault is initially set to 2, the system will be placed in the multi-
  user state via the /etc/rc2 procedure.

  Note that bcheckrc should always be executed before brc. Also, these shell
  procedures may be used for several run-level states.

SEE ALSO
  fsck(1M), init(1M), rc2(1M), shutdown(1M).
NAME
captinfo – convert a termcap description into a terminfo description

SYNOPSIS
captinfo [-v ...] [-V] [-1] [-w width] file ...

DESCRIPTION
captinfo looks in file for termcap descriptions. For each one found, an
equivalent terminfo(4) description is written to standard output, along with
any comments found. A description which is expressed as relative to
another description (as specified in the termcap tc= field) will be reduced to
the minimum superset before being output.

If no file is given, then the environment variable TERMCAPE is used for the
filename or entry. If TERMCAP is a full pathname to a file, only the termin-

nal whose name is specified in the environment variable TERM is extracted
from that file. If the environment variable TERMCAPE is not set, then the
file /etc/termcap is read.

-v
print out tracing information on standard error as the program
runs. Specifying additional -v options will cause more detailed
information to be printed.

-V
print out the version of the program in use on standard error
and exit.

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

-w
change the output to width characters.

FILES
/usr/lib/terminfo/* compiled terminal description database

CAVEATS

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

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

April 1990 - 1 - Version 5.0
The short two-letter name at the beginning of the list of names in a *termcap*
entry, a hold-over from an earlier version of the UNIX system, has been
removed.

**DIAGNOSTICS**

tgetent failed with return code n (reason).
   The termcap entry is not valid. In particular, check for an
   invalid 'tc=' entry.

unknown type given for the termcap code cc.
   The termcap description had an entry for cc whose type
   was not boolean, numeric or string.

wrong type given for the boolean (numeric, string) termcap code cc.
   The boolean *termcap* entry cc was entered as a numeric
   or string capability.

the boolean (numeric, string) termcap code cc is not a valid name.
   An unknown *termcap* code was specified.

tgetent failed on TERM=term.
   The terminal type specified could not be found in the
   *termcap* file.

TERM=term: cap cc (info ii) is NULL: REMOVED
   The *termcap* code was specified as a null string. The
   correct way to cancel an entry is with an '@', as in
   ':bs@:'. Giving a null string could cause incorrect
   assumptions to be made by the software which uses
   *termcap* or *terminfo*.

a function key for cc was specified, but it already has the value vv.
   When parsing the *ko* capability, the key cc was specified
   as having the same value as the capability cc, but the key
   cc already had a value assigned to it.

the unknown termcap name cc was specified in the *ko* termcap capability.
   A key was specified in the *ko* capability which could not
   be handled.

the vi character v (info ii) has the value xx, but ma gives n.
   The *ma* capability specified a function key with a value
   different from that specified in another setting of the same
   key.

the unknown vi key v was specified in the *ma* termcap capability.
   A *vi(1)* key unknown to *captoinfo* was specified in the *ma*
   capability.
Warning: `termcap sg (nn)` and `termcap ug (nn)` had different values.
   `terminfo` assumes that the `sg` (now `xmc`) and `ug` values were the same.

Warning: the string produced for `ii` may be inefficient.
   The parameterized string being created should be rewritten by hand.

Null termname given.
   The terminal type was null. This is given if the environment variable `TERM` is not set or is null.

cannot open `file` for reading.
   The specified file could not be opened.

SEE ALSO
   `infocmp(1M), tic(1M),
curses (3X), terminfo(4)` in the `Programmer's Reference Manual`
   Chapter 9 in the `Programmer's Guide`

NOTES
   `captoinfo` should be used to convert `termcap` entries to `terminfo(4)` entries because the `termcap` database (from earlier versions of UNIX System V) may not be supplied in future releases.
NAME
chkconfig – configuration state checker

SYNOPSIS
/etc/chkconfig [ flag [ on | off ]]

DESCRIPTION
chkconfig with no arguments, prints the state (on or off) of every
configuration flag found in the directory /etc/config. A flag is considered
on if its file contains the string “on” and off otherwise.

If flag is specified as the sole argument, chkconfig exits with status 0 if flag
is on and with status 1 if flag is off or nonexistent. The exit status can be
used by shell scripts to test the state of a flag. Here is an example using
sh(1) syntax:

    if /etc/chkconfig verbose; then
        echo "Verbose is on"
    else
        echo "Verbose is off"
    fi

The optional third argument allows the specified flag to be set.

These flags are used for determining the configuration status of the various
available subsystems and daemons during system startup and during system
operation.

FILES
/etc/config directory containing configuration flag files

SEE ALSO
cron(1M), rc0(1M), rc2(1M).
NAME
chroot – change root directory for a command

SYNOPSIS
/etc/chroot newroot command

DESCRIPTION
chroot causes the given command to be executed relative to the new root. The meaning of any initial slashes (/) in the path names is changed for the command and any of its child processes to newroot. Furthermore, upon execution, the initial working directory is newroot.

Notice, however, that if you redirect the output of the command to a file:

    chroot newroot command >x

will create the file x relative to the original root of the command, not the new one.

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

This command can be run only by the super-user.

CAVEAT
chroot has the side effect of changing the location of shared libraries. Shared libraries should exist in the new root for any commands that require them. For example, most commands are linked with the shared version of libc, so a copy of /lib/libc_s would be required in the new root in order for them to run.

SEE ALSO

BUGS
One should exercise extreme caution when referencing device files in the new root file system.
NAME

crthbl – generate character classification and conversion tables

SYNOPSIS

crthbl [file]

DESCRIPTION

The crthbl command creates a character classification table and an
upper/lower-case conversion table. The tables are contained in a byte-sized
array encoded such that a table lookup can be used to determine the charac-
ter classification of a character or to convert a character (see ctype(3C)).
The size of the array is 257*2 bytes: 257 bytes are required for the 8-bit
code set character classification table and 257 bytes for the upper- to
lower-case and lower- to upper-case conversion table.

crthbl reads the user-defined character classification and conversion infor-
mation from file and creates two output files in the current directory. One
output file, ctype.c (a C-language source file), contains the 257*2-byte
array generated from processing the information from file. You should
review the content of ctype.c to verify that the array is set up as you had
planned. (In addition, an application program could use ctype.c.) The first
257 bytes of the array in ctype.c are used for character classification.
The characters used for initializing these bytes of the array represent character
classifications that are defined in /usr/include/ctype.h; for example, _L
means a character is lower case and _S _B means the character is both a
spacing character and a blank. The last 257 bytes of the array are used for
character conversion. These bytes of the array are initialized so that charac-
ters for which you do not provide conversion information will be converted
to themselves. When you do provide conversion information, the first value
of the pair is stored where the second one would be stored normally, and
vice versa; for example, if you provide <0x41 0x61>, then 0x61 is stored
where 0x41 would be stored normally, and 0x61 is stored where 0x41
would be stored normally.

The second output file (a data file) contains the same information, but is
structured for efficient use by the character classification and conversion
routines (see ctype(3C)). The name of this output file is the value of the
character classification chrclass read in from file. This output file must be
installed in the /lib/chrclass directory under this name by someone who is
super-user or a member of group bin. This file must be readable by user,
group, and other; no other permissions should be set. To use the character
classification and conversion tables on this file, set the environmental vari-
able CHRCCLASS (see environ(5)) to the name of this file and export the
variable. For example, if the name of this file (and character class) is xyz,
sh(1) users should issue the commands:

   CHRCCLASS=xyz ; export CHRCCLASS

April 1990 - 1 -  Version 5.0
and `csh(1)` users should issue the command:

`setenv CHRCLASS xyz`

If no input file is given, or if the argument – is encountered, `chrtbl` reads from the standard input file.

The syntax of `file` allows the user to define the name of the data file created by `chrtbl`, the assignment of characters to character classifications and the relationship between upper- and lower-case letters. The character classifications recognized by `chrtbl` are:

- `chrclass` name of the data file to be created by `chrtbl`.
- `isupper` character codes to be classified as upper-case letters.
- `islower` character codes to be classified as lower-case letters.
- `isdigit` character codes to be classified as numeric.
- `isspace` character codes to be classified as a spacing (delimiter) character.
- `ispunct` character codes to be classified as a punctuation character.
- `isctl` character codes to be classified as a control character.
- `isblank` character code for the space character.
- `isxdigit` character codes to be classified as hexadecimal digits.
- `ul` relationship between upper- and lower-case characters.

Any lines with the number sign (`#`) in the first column are treated as comments and are ignored. Blank lines are also ignored.

A character can be represented as a hexadecimal or octal constant (for example, the letter `a` can be represented as `0x61` in hexadecimal or `0141` in octal). Hexadecimal and octal constants may be separated by one or more space and tab characters.

The dash character (`-`) may be used to indicate a range of consecutive numbers. Zero or more space characters may be used for separating the dash character from the numbers.

The backslash character (`\`) is used for line continuation. Only a carriage
return is permitted after the backslash character.

The relationship between upper- and lower-case letters (ul) is expressed as ordered pairs of octal or hexadecimal constants: `<upper-case_character lower-case_character>`. These two constants may be separated by one or more space characters. Zero or more space characters may be used for separating the angle brackets (< >) from the numbers.

**EXAMPLE**

The following is an example of an input file used to create the ASCII code set definition table on a file named ascii.

```
chrclass ascii
isupper 0x41 - 0x5a
islower 0x61 - 0x7a
isdigit 0x30 - 0x39
isspace 0x20 0x9 - 0xd
ispunct 0x21 - 0x2f  0x3a - 0x40  \ 0x5b - 0x60  0x7b - 0x7e
           0x21 - 0x2f  0x3a - 0x40  \ 0x5b - 0x60  0x7b - 0x7e
iscntrl 0x0 - 0x1f  0x7f
isblank 0x20
isxdigit 0x30 - 0x39  0x61 - 0x66  \ 0x41 - 0x46
              0x61 - 0x66  \ 0x41 - 0x46
ul  <0x41 0x61> <0x42 0x62> <0x43 0x63> \  <0x44 0x64> <0x45 0x65> <0x46 0x66> \  <0x44 0x64> <0x45 0x65> <0x46 0x66> \  <0x47 0x67> <0x48 0x68> <0x49 0x69> \  <0x4a 0x6a> <0x4b 0x6b> <0x4c 0x6c> \  <0x4d 0x6d> <0x4e 0x6e> <0x4f 0x6f> \  <0x50 0x70> <0x51 0x71> <0x52 0x72> \  <0x53 0x73> <0x54 0x74> <0x55 0x75> \  <0x56 0x76> <0x57 0x77> <0x58 0x78> \  <0x59 0x79> <0x5a 0x7a>
```

**FILES**

```
/lib/chrclass/* data file containing character classification and conversion tables created by chrtbl
/usr/include/ctype.h  header file containing information used by character classification and conversion routines
```

**SEE ALSO**

`environ(5)`,
`ctype(3C)` in the *Programmer's Reference Manual*. 

*April 1990* - 3 - Version 5.0
DIAGNOSTICS

The error messages produced by *chrtbl* are intended to be self-explanatory. They indicate errors in the command line or syntactic errors encountered within the input file.
NAME
ckbupscd – check file system backup schedule

SYNOPSIS
/etc/ckbupscd [ -m ]

DESCRIPTION
ckbupscd consults the file /etc/bupsched and prints the file system lists
from lines with date and time specifications matching the current time. If
the -m flag is present an introductory message in the output is suppressed
so that only the file system lists are printed. Entries in the /etc/bupsched file
are printed under the control of cron.

The System Administration commands bupsched/schedcheck are provided
to review and edit the /etc/bupsched file.

The file /etc/bupsched should contain lines of 4 or more fields, separated
by spaces or tabs. The first 3 fields (the schedule fields) specify a range of
dates and times. The rest of the fields constitute a list of names of file sys-
tems to be printed if ckbupscd is run at some time within the range given by
the schedule fields. The general format is:

time[,time] day[,day] month[,month] fsyslist

where:
time Specifies an hour of the day (0 through 23), matching any time
within that hour, or an exact time of day (0:00 through 23:59).
day Specifies a day of the week (sun through sat) or day of the month
(1 through 31).
month Specifies the month in which the time and day fields are valid.
Legal values are the month numbers (1 through 12).
fsyslist The rest of the line is taken to be a file system list to print.

Multiple time, day, and month specifications may be separated by commas,
in which case they are evaluated left to right.

An asterisk (*) always matches the current value for that field.

A line beginning with a sharp sign (#) is interpreted as a comment and
ignored.

The longest line allowed (including continuations) is 1024 characters.

EXAMPLES
The following are examples of lines which could appear in the
/etc/bupsched file.
06:00-09:00 fri 1,2,3,4,5,6,7,8,9,10,11 /appli
Prints the file system name /appli if ckbupscd is run between
6:00am and 9:00am any Friday during any month except
December.

00:00-06:00,16:00-23:59 1,2,3,4,5,6,7 1,8 /
Prints a reminder to backup the root (/) file system if ckbupscd is
run between the times of 4:00pm and 6:00am during the first
week of August or January.

FILES
/etc/bupsched specification file containing times and file system to back
up

SEE ALSO
cron(1M).

BUGS
ckbupscd will report file systems due for backup if invoked any time in the
window. It does not know that backups may have just been taken.
NAME
clri – clear i-node

SYNOPSIS
/etc/clri special i-number ...

DESCRIPTION
clri writes nulls on the inode table entry for i-number. This effectively
eliminates the i-node at that address. Special is the device name on which a
file system has been defined. After clri is executed, any blocks in the
affected file will show up as “not accounted for” when fsck(1M) is run
against the file-system. The i-node may be allocated to a new file.

Read and write permission is required on the specified special device.

This command is used to remove a file which appears in no directory; that
is, to get rid of a file which cannot be removed with the rm command.

SEE ALSO
fsck(1M), fsdb(1M), ncheck(1M).

WARNINGS
If the file is open for writing, clri will not work. The file system containing
the file should be NOT mounted.

If clri is used on the i-node number of a file that does appear in a directory,
it is imperative to remove the entry in the directory at once, since the i-node
may be allocated to a new file. The old directory entry, if not removed,
continues to point to the same file. This sounds like a link, but does not
work like one. Removing the old entry destroys the new file.
NAME
cron – clock daemon

SYNOPSIS
/etc/cron

DESCRIPTION
cron executes commands at specified dates and times. Regularly scheduled
commands can be specified according to instructions found in crontab files
in the directory /usr/spool/cron/crontabs. Users can submit their own
crontab file via the crontab(1) command. Commands which are to be exe-
cuted only once may be submitted via the at(1) command.

cron only examines crontab files and at command files during process ini-
tialization and when a file changes via crontab or at. This reduces the over-
head of checking for new or changed files at regularly scheduled intervals.
Since cron never exits, it should be executed only once. This is done rou-
tinely through /etc/rc2.d/S75cron at system boot time. /usr/lib/cron/FIFO
is used as a lock file to prevent the execution of more than one cron.

FILES
/usr/lib/cron main cron directory
/usr/lib/cron/FIFO used as a lock file
/usr/lib/cron/log accounting information
/usr/spool/cron spool area

SEE ALSO

DIAGNOSTICS
A history of all actions taken by cron are recorded in /usr/lib/cron/log.
NAME
dd – convert and copy a file

SYNOPSIS
dd [option=value] ...

DESCRIPTION
dd copies the specified input file to the specified output with possible conversions. The standard input and output are used by default. The input and output block size may be specified to take advantage of raw physical I/O.

option values
if=file input file name; standard input is default
of=file output file name; standard output is default
ibs=n input block size n bytes (default 512)
obs=n output block size (default 512)
bs=n set both input and output block size, superseding ibs and obs; also, if no conversion is specified, it is particularly efficient since no in-core copy need be done
cbs=n conversion buffer size
skip=n skip n input blocks before starting copy
seek=n seek n blocks from beginning of output file before copying
iseek=n seek n blocks from beginning of input file before copying
count=n copy only n input blocks
conv=asci convert EBCDIC to ASCII
ebcdic convert ASCII to EBCDIC
ibm slightly different map of ASCII to EBCDIC
lcase map alphabetics to lower case
ucase map alphabetics to upper case
swab swap every pair of bytes
noerror do not stop processing on an error
sync pad every input block to ibs
block convert ASCII to blocked ASCII
unblock convert blocked ASCII to ASCII

..., ... several comma-separated conversions

Where sizes are specified, a number of bytes is expected. A number may end with k, b, or w to specify multiplication by 1024, 512, or 2, respectively; a pair of numbers may be separated by x to indicate multiplication.

cbs is used only if conv=ascii or conv=ebcdic is specified. In the former case, cbs characters are placed into the conversion buffer (converted to ASCII). Trailing blanks are trimmed and a new-line added before sending the line to the output. In the latter case, ASCII characters are read into the
conversion buffer (converted to EBCDIC). Blanks are added to make up an output block of size cbs.

After completion, dd reports the number of whole and partial input and output blocks.

DIAGNOSTICS

\[ f+p \text{ blocks in(out)} \]  \hspace{1cm} \text{numbers of full and partial blocks read(written)}
NAME
devnm – device name

SYNOPSIS
/etc/devnm [ names ]

DESCRIPTION
devnm identifies the special file associated with the mounted file system
where the argument name resides.

This command is most commonly used by /etc/brc (see brc(1M)) to con-
struct a mount table entry for the root device.

EXAMPLE
The command:
    /etc/devnm /usr
produces
    /dev/dsk/ips0d0s2 usr
if /usr is mounted on /dev/dsk/ips0d0s2.

FILES
/dev/dsk/*
/etc/mtab

SEE ALSO
brc(1M).
NAME
dgld – Distributed Graphics Library server

SYNOPSIS
/usr/etc/dgld [-IM]

DESCRIPTION
Dgld is the server for programs linked with the Distributed Graphics Library. The server provides both a subprocess graphics facility and a networked graphics facility with an authentication process identical to rshd(1M). Dgld is started either by inetd(1M), dnserver, or as a forked child process depending on the type of DGL connection.

TCP socket connections are serviced by inetd. Inetd listens for connections on the port indicated in the “sgi-dgl” service specification; see services(4). When a connection is found, dgld is started as specified by the file /usr/etc/inetd.conf and given the socket. Authentication is performed using ruserok(3N) in the same manner as rshd and the server process’s group and user id are changed accordingly.

DECnet connections are serviced by dnserver which listens for connections for an object named “DGLD”. When a valid connection is established, dgld is started as specified in /usr/etc/dn/servers.reg and given the connection. Although a valid username and password is necessary to establish the connection, authentication using ruserok(3N) is still performed.

Local connections are processed by the client program forking a child process and calling exec(2) to run dgld. Two named pipes created in /tmp serve as the communication files between the client and server processes. No authentication is performed for local connections.

DIAGNOSTICS
Normally operation produces no diagnostic output unless an error occurs. All diagnostic messages are output to stderr. When dgld is started from inetd or dnserver both the -M and -I options are given. The -I option informs the graphics server that it was started from a network server process and enables output of all error messages to the system log maintained by syslogd(1M). The -M option disables all message output to stderr.

SEE ALSO
inetd (1M), rsh(1M), ruserok(3N), syslogd(1M)
NAME
diskusg – generate disk accounting data by user ID

SYNOPSIS
/usr/lib/acct/diskusg [options] [files]

DESCRIPTION
diskusg generates intermediate disk accounting information from data in files, or the standard input if omitted. diskusg output lines on the standard output, one per user, in the following format: uid login #blocks

where
uid the numerical user ID of the user.
login the login name of the user; and
#blocks the total number of disk blocks allocated to this user.

diskusg normally reads only the inodes of file systems for disk accounting. In this case, files are the special filenames of these devices.
diskusg recognizes the following options:

-s the input data is already in diskusg output format. diskusg combines all lines for a single user into a single line.
-v verbose. Print a list on standard error of all files that are charged to no one.
-i fnmlist ignore the data on those file systems whose file system name is in fnmlist. fnmlist is a list of file system names separated by commas or enclosed within quotes. diskusg compares each name in this list with the file system name stored in the volume ID (see labelit(1M)).
-p file use file as the name of the password file to generate login names. /etc/passwd is used by default.
-u file write records to file of files that are charged to no one. Records consist of the special file name, the inode number, and the user ID.

The output of diskusg is normally the input to acctdisk (see acct(1M)) which generates total accounting records that can be merged with other accounting records. diskusg is normally run in dodisk (see acctsh(1M)).

EXAMPLES
The following will generate daily disk accounting information for root on /dev/dsk/ips0d0s0:

/usr/lib/acct/diskusg /dev/dsk/ips0d0s0 | acctdisk > diskttacct
FILES
/etc/passwd used for user ID to login name conversions

SEE ALSO
acct(1M), acctsh(1M)
acct(4) in the *Programmer’s Reference Manual*
NAME
distcp – copy software distribution

SYNOPSIS
distcp [ -cnrv ] from to

DESCRIPTION

distcp copies all or part of a distribution from one location to another. It can also compare two distributions or parts of a distribution. Options:

-c Compare from and to rather than copying.
-s Compare silently; exit status only.
-v Verbose; report names as files are copied.
-n No standalone files sa and mr.
-r Retension tape before reading or writing.

A software product can reside on magnetic tape or in disk files. A product named foo will consist several physical files. The file by which the product is identified is called a product descriptor and is named foo. The other files that make up the product are called foo.idb, and foo.image for each image in the product.

There are two additional files that are used along with the product files. One of these is called sa, and contains the standalone tools used during software installation. The other is called mr, and may or may not contain additional components of the standalone environment. Compatibility issues require its presence even if it is empty.

The integrity of the software distribution will be lost if any of these files are altered in any way.

The from and to arguments indicate where to copy the distribution from and to. They can be a tape device, the pathname of a product descriptor file, or the pathname of a directory containing several products. Any of these names may be prefixed with hostname: or user@hostname: to access a remote machine.

The two standalone files sa and mr are included in every copy, unless the -n flag is given.

It is not possible to copy more than one product to tape, therefore also not possible read more than one product from tape.

A collection of software products copied to a disk directory, along with a single copy of the standalone files sa and mr, will make a convenient distribution source for the inst command in a network environment.
SEE ALSO

inst(1m), versions(1m).
NAME
du – summarize disk usage

SYNOPSIS
du [ -sarkIL ] [ names ]

DESCRIPTION

Du reports the number of data blocks contained in all files and (recursively) directories within each directory and file specified by the names argument. By default, the block counts reported are in terms of 512 byte blocks, but this can be modified to 1024 byte blocks by specifying the -k option (see below). The block count includes only the actual data blocks used by each file and directory, not indirect blocks or other file system data structures required to represent the file. Du keeps track of files that have more than one name (hard link) and counts the blocks used by the file only the first time it finds a link to the file.

If the names argument is missing, the current working directory is used.

The following options are accepted:

-s causes only the grand total (for each of the specified names) to be given.

-a causes an output line to be generated for each file.

If neither -s or -a is specified, an output line is generated for each directory only.

-r will cause du to generate messages about directories that cannot be be read, files that cannot be opened, etc., rather than being silent (the default).

-k will cause du to express all block counts in terms of 1024 byte blocks, instead of the default 512 byte blocks. Note that the internal accounting is still done in terms of 512 bytes blocks, since that's the actual file system allocation granularity. The numbers are just converted to 1024 blocks before being printed. Note that this means that the total block count shown for a directory will sometimes be less than the sum of the block counts of the contained files and directories (when more than one of the contained files is less than 512 bytes in size).

-l will prevent du from descending into and computing disk usage for directories and files that do not reside on local disk file systems.

-L will cause du to follow symbolic links. The default behavior is not to follow symbolic links, so that du normally reports the size of the symbolic link itself. When -L is specified, du will report the size of the target of the symbolic link. If the symbolic link points to a
directory when \(-L\) is specified, \textit{du} will recursively descend the directory just as it does for normal directories.

**BUGS**

If the \(-a\) option is not used, non-directories given as arguments are not listed.

When using the \(-l\) option, non-local directories and files given as arguments are not listed.
NAME
dvhtool — modify and obtain disk volume header information

SYNOPSIS

/etc/dvhtool [-b bootfile ] [-v [add unix_file dvh_file]
[creat unix_file dvh_file] [get dvh_file unix_file]
[delete dvh_file] [list]] [header_filename]

DESCRIPTION

Dvhtool allows modification of the disk volume header information, a block
located at the beginning of all disk media. The disk volume header consists
of three main parts: the device parameters, the partition table, and the
volume directory. The volume directory is used to locate files kept in the
volume header area of the disk for standalone use. The partition table
describes the logical device partitions. The device parameters describe the
specifics of a particular disk drive.

Note that it is necessary to be superuser in order to use dvhtool.

Invoked with no arguments (or just a volume header name), dvhtool allows
the user to interactively examine and modify the disk volume header on the
root drive. The read command prompts for the name of the device file for
the volume header to be worked on. This may be /dev/rvh for the header of
the root disk, or the header name of another disk in the /dev/disk directory,
see vh(7m). It then reads the volume header from the specified device.
The ve, pt, and dp commands first list their respective portions of the
volume header and then prompt for modifications. The write command
writes the possibly modified volume header to the device.

NOTE: use of dvhtool for changing partitions and parameters is not recom-
mented. Parameters and partitions should be manipulated with fx(1m).

COMMAND LINE ARGUMENTS

Invoked with arguments, dvhtool reads the volume header, performs the
specified operations, and then writes the volume header. If no
header_filename is specified on the command line, /dev/rvh is used.

The following describes dvhtool's command line arguments.

The -v flag provides four options for modifying the volume directory infor-
information in the disk volume header. The creat option allows creation of a
volume directory entry with the name dvh_file and the contents of unix_file.
If an entry already exists with the name dvh_file, it is overwritten with the
new contents. The add option adds a volume directory entry with the name
dvh_file and the contents of unix_file. Unlike the creat option, the add
options will not overwrite an existing entry. The delete option removes the
entry named dvh_file, if it exists, from the volume directory. The get option
copies the requested file from the volume header to the filesystem. The list
option lists the current volume directory contents.

SEE ALSO
vh(7m)
f(1m)

NOTE
May require several Mbytes of disk space in the /tmp directory when creating or adding files if the free space in the volume header is fragmented. This also makes dvhtool run much slower, since all files must then be copied to /tmp, and then back to the volume header.
NAME
fingerd – remote user information server

SYNOPSIS
/usr/etc/fingerd [-S]

DESCRIPTION
Fingerd is a simple protocol based on RFC742 that provides an interface to
the Name and Finger programs at several network sites. The program is
supposed to return a friendly, human-oriented status report on a particular
person. There is no required format and the protocol consists mostly of
specifying a single "command line".

Fingerd listens for TCP requests at port 79. Once connected, it reads a sin-
gle line terminated by a <CRLF> and passes the first three words on the line
as arguments to finger(1). Fingerd closes its connection as soon as the out-
put is finished. It can be invoked at a remote site using the finger command
by finger user@remote or finger @remote. The -S option causes fingerd
to suppress information about login status, home directory, and shell, which
might be used to attack security.

SEE ALSO
finger(1), inetd(1M), telnet(1C)

BUGS
Connecting directly to the server from a TIP or an equally narrow-minded
TELNET-protocol user program can result in meaningless attempts at
option negotiation being sent to the server, which will foul up the command
line interpretation. Fingerd should be taught to filter out IAC’s and perhaps
even respond negatively (IAC WON’T) to all option commands received.
NAME
fsck, dfsck – check and repair file systems

SYNOPSIS
/etc/fsck [-y] [-n] [-t file] [-q] [-D] [-f] [ file-systems ]
/etc/dfsck [options1] fsys1 ... - [options2] fsys2 ...

DESCRIPTION
Fsck

fsck audits and interactively repairs inconsistent conditions for file systems. If the file system is found to be consistent, the number of files, blocks used, and blocks free are reported. If the file system is inconsistent the user is prompted for concurrence before each correction is attempted. It should be noted that most corrective actions will result in some loss of data. The amount and severity of data loss may be determined from the diagnostic output. The default action for each correction is to wait for the user to respond yes or no. If the user does not have write permission fsck defaults to a -n action.

The following options are accepted by fsck.

-y Assume a yes response to all questions asked by fsck.
-n Assume a no response to all questions asked by fsck; do not open the file system for writing.
-t If fsck cannot obtain enough memory to keep its tables, it uses a scratch file. If the -t option is specified, the file named in the next argument is used as the scratch file, if needed. Without the -t flag, fsck will prompt the user for the name of the scratch file. The file chosen should not be on the file system being checked, and if it is not a special file or did not already exist, it is removed when fsck completes.
-q Quiet fsck. Do not print size-check messages. Unreferenced fifos will silently be removed. If fsck requires it, counts in the superblock will be automatically fixed and the free list salvaged.
-D Directories are checked for bad blocks. Useful after system crashes.
-f Fast check. Check block and sizes and check the free list. The free list will be reconstructed if it is necessary.

If no file-systems are specified, fsck will read a list of default file systems from the file /etc/fstab. Note that this will not include the root filesystem, fsck will run on root only if this is explicitly specified.
Normally, a filesystem must be unmounted in order to run *fsck* on it, an error message is printed and no action taken if invoked on a mounted filesystem. The one exception to this is the root filesystem, which obviously must be mounted in order to run *fsck*. If inconsistencies are detected when running on root, *fsck* causes a remount of root.

Inconsistencies checked are as follows:

1. Blocks claimed by more than one i-node or the free list.
2. Blocks claimed by an i-node or the free list outside the range of the file system.
3. Incorrect link counts.
4. Size checks:
   - Incorrect number of blocks.
   - Directory size not 16-byte aligned.
5. Bad i-node format.
6. Blocks not accounted for anywhere.
7. Directory checks:
   - File pointing to unallocated i-node.
   - I-node number out of range.
8. Super Block checks:
   - More blocks for i-nodes than there are in the file system.
9. Bad free block list/bitmap format.
10. Total free block and/or free i-node count incorrect.

Orphaned files and directories (allocated but unreferenced) are, with the user's concurrence, reconnected by placing them in the *lost+found* directory, if the files are nonempty. The user will be notified if the file or directory is empty or not. Empty files or directories are removed, as long as the *-n* option is not specified. *fsck* will force the reconnection of nonempty directories. The name assigned is the i-node number. The only restriction is that the directory *lost+found* must preexist in the root of the file system being checked and must have empty slots in which entries can be made. This is accomplished by making *lost+found*, copying a number of files to the directory, and then removing them (before *fsck* is executed).

Checking the raw device is almost always faster and should be used with everything but the root file system.

*Dfsck*

*Dfsck* should not be used to check the root file system.

*Dfsck* allows two file system checks on two different drives simultaneously. *options1* and *options2* are used to pass options to *fsck* for the two sets of file systems. A – is the separator between the file system groups.
The `dfsck` program permits a user to interact with two `fsck` programs at once. To aid in this, `dfsck` will print the file system name for each message to the user. When answering a question from `dfsck`, the user must prefix the response with a `1` or a `2` (indicating that the answer refers to the first or second file system group).

**SUPERBLOCKS AND FILESYSTEM ROBUSTNESS**

In IRIX 3.3 a replicated superblock has been added to the efs filesystem, situated at the end of the filesystem space. If `fsck` cannot read the primary superblock it will attempt to use the replicated superblock, printing a message to notify the user of the situation. This is automatic; no user intervention is required. Further, `fsck` attempts to determine if a replicated superblock exists, and if not will optionally create one; thus allowing older filesystems to benefit from this feature.

Finally, if no superblock can be found on a damaged filesystem, it may be possible to regenerate one by using the new `-r` option of `mkfs(1M)`, and then use `fsck` to salvage the filesystem.

**FILES**

`/etc/fstab` contains default list of file systems to check.

**SEE ALSO**

`mkfs(1M)`, `ncheck(1M)`,
`uadmin(2)`, `fstab(4)`, `fs(4)` in the *Programmer’s Reference Manual*.

**BUGS**

I-node numbers for `.` and `..` in each directory are not checked for validity.
NAME
fsstat – report file system status

SYNOPSIS
/etc/fsstat special_file

DESCRIPTION
fsstat reports on the status of the file system on special_file. During startup, this command is used to determine if the file system needs checking before it is mounted. fsstat succeeds if the file system is unmounted and appears okay. For the root file system, it succeeds if the file system is active and not marked bad.

SEE ALSO

DIAGNOSTICS
The command has the following exit codes:

0 -- the file system is not mounted and appears okay,
   (except for root where 0 means mounted and okay).
1 -- the file system is not mounted and needs to be checked.
2 -- the file system is mounted.
3 -- the command failed.
NAME
  fstyp – determine file system identifier

SYNOPSIS
  fstyp special

DESCRIPTION
  fstyp allows the user to determine the file system identifier of mounted or
  unmounted file systems using heuristic programs. The file system type is
  required by mount(2) and sometimes by mount(1M) to mount file systems
  of different types.

  The directory /etc/fstyp.d contains a program for each file system type to be
  checked; each of these programs applies some appropriate heuristic to
determine whether the supplied special file is of the type for which it
checks. If it is, the program prints on standard output the usual file-system
identifier for that type and exits with a return code of 0; otherwise it prints
error messages on standard error and exits with a non-zero return code.

  fstyp runs the programs in /etc/fstyp.d in alphabetical order, passing special
  as an argument; if any program succeeds, its file-system type identifier is
  printed and fstyp exits immediately. If no program succeeds, fstyp prints
  "Unknown_fstyp" to indicate failure.

WARNING
  The use of heuristics implies that the result of fstyp is not guaranteed to be
  accurate.

SEE ALSO
  mount(1M),
NAME
ftp - Internet File Transfer Protocol server

SYNOPSIS
/usr/etc/ftp [-d] [-l] [-t timeout] [-T maxtimeout] [-u umask]

DESCRIPTION
_Ftpd_ is the Internet File Transfer Protocol server process. The server uses
the TCP protocol and listens at the well-known port specified in the _services(4)_ file.

_Ftpd_ is started by _ineted(1M)_ whenever a remote client makes a connection
request. The following options should specified on the ftpd line in _ineted's_
configuration file, /usr/etc/inetd.conf.

If the `-d` option is specified, debugging information is written to the syslog.
If the `-l` option is specified, each ftp session login is logged in the syslog.

The ftp server will timeout an inactive session after 15 minutes. If the `-t`
option is specified, the inactivity timeout period will be set to _timeout_
seconds. A client may also request a different timeout period; the max-
umum period allowed may be set to _timeout_ seconds with the `-T` option.
The default limit is 2 hours.

The `-u` option specifies the default file protection mask (see _umask(1)_). If
not specified, the default mask is set to 022 (group- and world-readable). A
client may also request a different mask.

If the file _/etc/issue_ exists, _ftp_ prints it before issuing the "ready" mes-

sage.

The ftp server currently supports the following ftp requests; case is not dis-
tinguished.

<table>
<thead>
<tr>
<th>Request</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABOR</td>
<td>abort previous command</td>
</tr>
<tr>
<td>ACCT</td>
<td>specify account (ignored)</td>
</tr>
<tr>
<td>ALLO</td>
<td>allocate storage (vacuously)</td>
</tr>
<tr>
<td>APPE</td>
<td>append to a file</td>
</tr>
<tr>
<td>CDUP</td>
<td>change to parent of current working directory</td>
</tr>
<tr>
<td>CWD</td>
<td>change working directory</td>
</tr>
<tr>
<td>DELE</td>
<td>delete a file</td>
</tr>
<tr>
<td>HELP</td>
<td>give help information</td>
</tr>
<tr>
<td>LIST</td>
<td>give list files in a directory (&quot;ls -1A&quot;)</td>
</tr>
<tr>
<td>MKD</td>
<td>make a directory</td>
</tr>
<tr>
<td>MDTM</td>
<td>show last modification time of file</td>
</tr>
<tr>
<td>MODE</td>
<td>specify data transfer mode</td>
</tr>
<tr>
<td>NLST</td>
<td>give name list of files in directory</td>
</tr>
<tr>
<td>NOOP</td>
<td>do nothing</td>
</tr>
</tbody>
</table>
PASS        specify password
PASV        prepare for server-to-server transfer
PORT        specify data connection port
PWD         print the current working directory
QUIT        terminate session
REST        restart incomplete transfer
RETR        retrieve a file
RMD         remove a directory
RNFR        specify rename-from file name
RNTO        specify rename-to file name
SITE        non-standard commands (see next section)
SIZE        return size of file
STAT        return status of server
STOR        store a file
STOU        store a file with a unique name
STRU        specify data transfer structure
SYST        show operating system type of server system
TYPE        specify data transfer type
USER        specify user name
XCUP        change to parent of current working directory (deprecated)
XCWD        change working directory (deprecated)
XMKD        make a directory (deprecated)
XPWD        print the current working directory (deprecated)
XRMD        remove a directory (deprecated)

The following non-standard or UNIX specific commands are supported by the SITE request.

Request   Description
UMASK      change umask, e.g., SITE UMASK 002
IDLE       set idle-timer, e.g., SITE IDLE 60
CHMOD      change mode of a file, e.g., SITE CHMOD 755 filename
HELP       give help information, e.g., SITE HELP

The remaining ftp requests specified in Internet RFC 959 are recognized, but not implemented. MDTM and SIZE are not specified in RFC 959, but will appear in the next updated FTP RFC.

The ftp server will abort an active file transfer only when the ABOR command is preceded by a Telnet "Interrupt Process" (IP) signal and a Telnet "Synch" signal in the command Telnet stream, as described in Internet RFC 959. If a STAT command is received during a data transfer, preceded by a Telnet IP and Synch, transfer status will be returned.
**Ftpd** interprets file names according to the "globbing" conventions used by *csh*(1). This allows users to utilize the metacharacters "*?[]{}". *Ftpd* authenticates users according to three rules.

1) The user name must be in the password data base, *etc/passwd*, and not have a null password. In this case a password must be provided by the client before any file operations may be performed.

2) The user name must not appear in the file *etc/ftpusers*. However, if the user name is in *etc/ftpusers* followed by the white-space separated keyword "restrict", the user is allowed restricted access privileges, as described below.

3) If the user name is "anonymous" or "ftp", an anonymous ftp account must be present in the password file (user "ftp"). In this case the user is allowed to log in by specifying any password (by convention this is given as the client user and host name).

For the restricted and anonymous accounts, *ftp* takes special measures to restrict the client's access privileges. The server performs a *chroot*(2) command to the home directory of the user and prints the file README if it exists. In order that system security is not breached, it is recommended that the subtree be constructed with care; the following rules are recommended, using the "ftp" anonymous account as an example.

```
~ftp  Make the home directory owned by "ftp" and unwritable by anyone.

~ftp/bin  Make this directory owned by the super-user and unwritable by anyone. The program *ls*(1) must be present to support the list commands. This program should have mode 111 (see *chmod*(1)).

~ftp/etc  Make this directory owned by the super-user and unwritable by anyone. The files *passwd*(4) and *group*(4) must be present for the *ls* command to be able to produce owner names rather than numbers. The password field in *passwd* is not used, and should not contain real encrypted passwords. These files should be mode 444.

~ftp/pub  Make this directory owned by "ftp". If local users and remote anonymous users are allowed to write in this directory, change the directory’s mode to 777. Users can then place files which are to be accessible via the anonymous account in this directory. If write accesses are to be denied, change the directory’s mode to 555.
```
FILES
/etc/ftpusers  list of unwelcome/restricted users
/etc/issue    welcome notice

SEE ALSO
ftp(1C), inetd(1M), syslogd(1M)

BUGS
The anonymous account is inherently dangerous and should avoided when possible.

The server must run as the super-user to create sockets with privileged port numbers. It maintains an effective user id of the logged in user, reverting to the super-user only when binding addresses to sockets. The possible security holes have been extensively scrutinized, but are possibly incomplete.
NAME

fuser – identify processes using a file or file structure

SYNOPSIS

/etc/fuser [–ku] files [–] [ [–ku] files ]

DESCRIPTION

fuser outputs the process IDs of the processes that are using the files specified as arguments. Each process ID may be followed by a letter code, interpreted as follows: if the process is using the file as its current directory, the code is c; if the file is the process’s root directory, the code is r; if the file is in execution, the code is x. No code means the process is holding the file open. If the file is being used as the process accounting file, fuser prints acct instead of a process ID and letter code.

If file names a block special device containing a mounted file system, all processes using any file on that device are listed. If file has the form hostname:pathname and names a mounted NFS filesystem, all processes using any file in that system are listed.

The following options may be used with fuser:

–u the user login name, in parentheses, also follows the process ID.

–k the SIGKILL signal is sent to each process. Since this option spawns kills for each process, the kill messages may not show up immediately [see kill(2)].

If more than one group of files are specified, the options may be respecified for each additional group of files. A lone dash cancels the options currently in force; then, the new set of options applies to the next group of files.

The process IDs are printed as a single line on the standard output, separated by spaces and terminated with a single new line. All other output is written on standard error.

Any user can use fuser. Only the super-user can terminate another user’s process.

FILES

/unix for the system namelist
/dev/kmem the system memory image

SEE ALSO

mount(1M),
NAME

fwtmp, wtmpfix – manipulate connect accounting records

SYNOPSIS

/usr/lib/acct/fwtmp [-ic]
/usr/lib/acct/wtmpfix [files]

DESCRIPTION

fwtmp reads from the standard input and writes to the standard output, converting binary records of the type found in wtmp to formatted ASCII records. The ASCII version is useful to enable editing, via ed(1), bad records or general purpose maintenance of the file.

The argument -ic is used to denote that input is in ASCII form, and output is to be written in binary form.

wtmpfix examines the standard input or named files in wtmp format, corrects the time/date stamps to make the entries consistent, and writes to the standard output. A - can be used in place of files to indicate the standard input. If time/date corrections are not performed, acctcon(1) will fault when it encounters certain date-change records.

Each time the date is set, a pair of date change records are written to /etc/wtmp. The first record is the old date denoted by the string old time placed in the line field and the flag OLD_TIME placed in the type field of the <utmp.h> structure. The second record specifies the new date and is denoted by the string new time placed in the line field and the flag NEW_TIME placed in the type field. wttmpfix uses these records to synchronize all time stamps in the file.

In addition to correcting time/date stamps, wttmpfix will check the validity of the name field to ensure that it consists solely of alphanumeric characters or spaces. If it encounters a name that is considered invalid, it will change the login name to INVALID and write a diagnostic to the standard error. In this way, wttmpfix reduces the chance that acctcon(1) will fail when processing connect accounting records.

FILES

/etc/wtmp
/usr/include/utmp.h

SEE ALSO

acct(1M), acctcms(1M), acctcon(1M), acctmerg(1M), acctprc(1M), acctsh(1M), runacct(1M)
acctcom(1), ed(1) in the User’s Reference Manual
NAME
fx – disk utility

SYNOPSIS
fx [ -x ] [ -r retries ] [ -l logfile ] ["drive_spec" ["drive_type"]]

DESCRIPTION
fx is an interactive, menu-driven disk utility. It allows bad blocks on a disk to be detected and mapped out. It also allows display of information stored on the label of the disk, including partition sizes, disk drive parameters and the volume directory.

An extended mode, available by invoking with the -x flag, provides additional functions normally used during factory set-up or servicing of disks, such as formatting the disk and creating or modifying the disk label. **Warning:** unless you are very familiar with the parameters and partitions of your disks, you are strongly advised not to invoke the extended mode of fx. A mistake in extended mode can destroy all the data on the disk.

The -r retries option allows you to specify how many retries fx will attempt when exercising the disk. If you have persistent soft errors, -r 0 will usually allow fx to find the bad sectors and spare them.

The -l logfile option (in the unix command version only) will cause fx to log disk errors, blocks that are forwarded, and other severe errors in the given file.

USING FX
fx exists in both standalone and command versions. A copy of the standalone version is normally kept in /stand/fx, and may be invoked when the system is not running by means of the standalone shell sash. This is described in more detail in the user’s guide chapter on fx.

The command version of fx is simply invoked by name like any regular command. Note that it is necessary to be superuser to use it since it accesses the device files for the disks.

On invocation, fx prompts for a disk controller type, with a default of the root disk controller type. Recognized controller types are dksc for SCSI drives, dkip for ESDI drives with Interphase controllers, xyl for SMD drives with Xylogics controllers, and fd for floppy drives (command version only).

fx Controller number will normally be 0 unless your system has more than one controller and you wish to work on disks attached to the second controller. Drive number depends on controller type. ESDI and SMD drives are numbered from 0 to 1 (or 0 to 3, if the controller can handle 4 drives), with drive 0 on controller 0 normally used as the root disk. SCSI drives are numbered from 1 to 6, with drive 1 on controller 0 normally used as the root
disk. \texttt{fx} next prompts for the drive type, with a default of the drive type stored in the disk label.

The controller type, controller number and drive number as well as drive type may be given as command line parameters, bypassing the interactive questions just described. The format is:

\texttt{fx "controller	exttt{type} (controller	exttt{number}, drive	exttt{number})" "drive	exttt{type}"}

for example:

\texttt{fx "dkip(0,1)" "Hitachi 512-17"}

The quotes are essential in the first argument since parentheses are shell special characters, and in the second because the drive name contains a space. For floppy disk drives, you are also prompted for the density to use.

Once controller type, controller number, drive number and drive type are selected, \texttt{fx} performs some sanity checks in regards to partition layout, etc. If any 'major' differences are found, you will be asked whether you want to use the existing values. It is almost always correct to keep the existing values, unless you are going to initialize the disk anyway.

\texttt{fx then} enters its main menu. Menu items may be selected either by the \texttt{name} or \texttt{number} shown on the menu. A menu item may be an action (e.g \texttt{exit}), or the name of a submenu (e.g. \texttt{badblock}).

Selecting a submenu name will cause that submenu to be displayed, and items from it may then be selected.

To return to a parent menu from a submenu, enter two dots ("..").

To obtain a "help" display giving more information about the items on the current menu, enter a question mark ("?") at the prompt. Many of the functions listed below have options to modify their actions; to obtain information about them, enter "? item" where the item may be either the name or the number.

To exit from \texttt{fx}, select \texttt{exit} at the main menu. Selecting ".." at the top level allows you to select a different disk without having to exit and restart.

Once the main menu is reached, \texttt{fx} catches interrupts: an interrupt will stop any operation in progress but will not terminate \texttt{fx} itself.

\textbf{BADBLOCK MANAGEMENT}

Most disks have a number of defective spots where data cannot be stored. The disk controller is able to get around this by dynamically replacing a bad block with a good block from a pool reserved for this purpose. A list is kept on the disk of defects and their replacements. If new bad blocks develop
during the life of the system, it is necessary to add these new bad blocks to
the badblock list. Typically, the disk driver will print error messages on the
console when it encounters a bad block. These error messages will give the
location of the bad block, either as a single block number or as cylinder,
head and sector (in a form such as chs: 123/4/5) depending on the controller
type. The disk is identified by its special file name: see dkip(7), dksc(7) or
xyl(7).

Note that the SCSI disk driver prints bad block numbers relative to the start
of the partition it is accessing. In this case, it is therefore necessary to add
the starting blocknumber of the partition to the blocknumber printed by the
driver in order to arrive at an absolute blocknumber before adding it as a
bad block.

Warning: for ESDI and SMD drives, fx attempts to save data when mapping
out bad blocks. Data is NOT saved for SCSI disks. In all cases it is strongly
recommended to make a backup of the disk before proceeding with any
badblock operations. Badblock mapping is NOT supported for floppy disk
drives.

To map out a bad block, invoke fx, select the appropriate disk and go to the
badblock menu. For SMD or ESDI disks, select the readinbb item to read in
the existing badblock list from the disk. Next, select the showbb item to
display the existing badblock list. Typically, there will be a small number of
entries. If there are no entries, it is possible that the badblock list has been
lost or corrupted. In this case, for ESDI or SMD drives, you should attempt
to recover the manufacturer’s original defect list by entering readdefects.

Note that no corresponding function exists for SCSI drives, and the reade-
defects menu item does not appear. Also, for SCSI drives it is not necessary to
read a badblock list before adding new bad blocks.

To enter new bad blocks, select the addbb item. Then enter the location of
the bad block (fx accepts either a single blocknumber or a
cylinder/head/sector specification). More than one badblock can be
entered; when you have finished entering, terminate the entries by entering
two dots ("." ). For SMD and ESDI disks, the updated badblock list must
then be saved to disk and the new bad blocks mapped out. Select the for-
ward option on the badblock menu to do this.
For SCSI disks, bad blocks are mapped out as soon as they are entered via
the addbb function, and nothing more need be done; the forward item does
not appear on the menu.
It is also possible to scan the disk surface for bad blocks automatically. To do this, select the `exercise` option on the main `fx` menu. Select `sequential` on the `exercise` menu and use the defaults for exercise type and range: this will cause a read-only scan of the entire disk surface. Defects detected during the scan will be automatically added to the badblock list. Note that all exercises in the normal (non-extended) mode of `fx` are read-only for safety; this means that some defects may escape detection. Also, the exercise function will mark as bad only unrecoverable blocks: blocks which can be accessed but need one or more retries will not be marked as bad since disk drivers hide soft errors from user programs. For this reason, it is advisable to keep notes of persistent soft error messages (retries) which the disk driver prints on the console during normal operations, and add these bad blocks manually. It is best to replace a block which is going bad before it becomes unreadable.

**FX DISPLAY FUNCTIONS**

`fx` can display the information in the various parts of the disk label. To do this, select the `label` option at the main menu. Then select the `readin` function, and select the part(s) of the label you wish to display; this will read in the information from the disk. Return to the `label` menu and select `show`; the various parts of the label can then be selected for display.

**FX DEBUG FUNCTIONS**

`fx` has a menu of disk debug functions. For safety reasons, these are somewhat restricted in the normal (non-extended) mode. However, a function which may be useful is the ability to directly read and display the contents of any block on the disk. In the `debug` menu, select `seek`. You can then enter the blocknumber you wish to read, either as a single blocknumber or as a cylinder/head/sector specification. Note that this is an absolute block number, starting at the beginning of the disk: if you wish to read the Nth block of a given partition you must convert this to an absolute block number by adding the starting block of the partition. (Partition information may be obtained by the label display function mentioned above).

Once the block is selected, it can be read by the `readbuf` item. (This can read up to 100 consecutive blocks from the selected starting block). Data read from the disk can then be displayed by `dumpbuf`; it is shown in both hex and (if printable) character format.

**MENU DESCRIPTIONS**

The top level `fx` menu contains the following choices:

1) `exit` Exits from `fx`. If changes have been made to the copy `fx` keeps of the disk label and this has not been written to the disk, a prompt will give the option to write it to disk.
2) badblock
   Selects the menu of operations dealing with bad block handling.

3) debug
   Selects the menu of debug functions.

4) exercise
   Selects the menu of functions for analysing the disk surface to find bad blocks.

5) label
   Selects the menu of functions for reading (and, in extended mode, modifying) the disk label.

   The remaining options appear only in extended mode.

6) auto
   A function for initializing a new disk. The disk is formatted, a label is created and written to it, and it is exercised in order to detect and map out bad blocks.

7) format
   Formats the disk. With SCSI disks, the whole disk is formatted. With ESDI or SMD disks, it is possible to format a range of cylinders; prompts are given for start cylinder and number of cylinders, the defaults being the entire disk. Note that with SMD disks, the defect information placed on the disk by the manufacturer is destroyed by formatting, so when formatting an SMD disk, fx automatically attempts to read and save this information first. If the disk has been previously formatted, fx will find no defect information on the disk. In this case, it will print a message saying that no defect information was found after trying a few tracks, and offering the possibility of abandoning the search for it or continuing. If it is known that an SMD disk has been previously formatted, the reformat will be considerably speeded up by a "no" answer to this question.

8) restore
   This function allows a UNIX file to be copied to a partition of the disk. By selecting as source a device special file of a partition on another disk, this function allows disks to be cloned.

BADBLOCK MENU

Badblocks are handled differently by SCSI and ESDI/SMD controllers. In the case of SCSI controllers, the list of badblocks is maintained by the SCSI controller/formatter hardware; it can be interrogated and altered but does not appear in the user-readable part of the disk. For ESDI/SMD disks, the badblock list is a structure maintained explicitly by fx in the SGI-specific label area on the disk. ESDI/SMD badblocks are managed on a track basis: if a track has one or more bad blocks, the entire track is replaced with one
from the "track replacement" area reserved on the disk.

The badblock menu contains the following choices:

1) **addbb**

Allows new bad blocks to be added to the badblock list. Blocks may be identified either by a single blocknumber or as cylinder/head/sector. To terminate adding bad blocks, enter two dots (".."); this returns to the badblock menu. In the SCSI case, an entered bad block is immediately inserted in the on-disk list maintained by the SCSI controller/formatter. In the ESDI/SMD case, it goes into the in-core copy of the badblock list maintained by fx, and does not become effective until the forward command has been given.

2) **deletebb**

Deletes a block from the badblock list. Essentially this is just to allow any mistake made during entry to be corrected; once a block has been identified as bad it will remain so. Note that this function does not appear for SCSI disks; there is no way to remove an added badblock from a SCSI disk without reformatting the whole disk.

3) **readdefects**

This function appears in the menu only for ESDI or SMD disk types. It reads the defect information placed on the disk by the disk manufacturer, and adds bad blocks to the bad block list. (If the blocks identified by the manufacturer’s defect information are already in the bad block list, nothing is altered). This function is useful for retrieving original defect information if the badblock list in the disk label has become lost or corrupted for some reason. Note that once an SMD disk has been formatted, the manufacturer’s defect information is overwritten.

4) **readinbb**

Reads the bad block list in the disk label into memory. (Note that fx does not keep an in-core bad block list for SCSI drives, so this function appears only for ESDI or SMD drive types). It should be used before adding any bad blocks, to ensure that the current badblock list is being worked on.

5) **showbb**

This displays the current badblock list. For ESDI and SMD disks, the displayed list is the in-core copy kept by fx, originally read in with readinbb and possibly modified with addbb.

For SCSI disks, it is obtained by interrogating the SCSI controller. What is displayed in this case is the physical location of the bad
sectors; the SCSI controller does not retain a record of the former logical block number of a bad block.

6) *forward*

For ESDI and SMD disks, this saves the badblock list to disk, and causes the disk controller to map out any newly added bad blocks. (It does not appear for SCSI disks since added bad blocks on a SCSI disk are effective immediately).

The remaining option appears only in extended mode.

7) *createbb*

This clears out any existing badblock list held by *fx* for the disk. It appears of course only for ESDI or SMD disks, and would normally be used only when a disk is being completely reformatted.

**DEBUG MENU**

This gives access to miscellaneous debug functions, mostly for reading and writing disk blocks. An internal memory buffer is provided as a source or destination for data; the contents of this buffer may be displayed and edited. In the normal (non-extended) mode of *fx*, only non-destructive functions are available. In the extended mode, disk blocks may be written as well as read. Options are:

1) *cmpbuf*

   This allows blocks of data in different areas of the buffer to be compared, for example, to compare written and read-back data. It prompts for the starts of the two areas to be compared (relative to the beginning of the internal buffer), and for the length of comparison.

2) *dumpbuf*

   This allows display of the contents of the buffer. It prompts for start address (relative to beginning of buffer), length to display and display format: bytes, (2-byte) words, or (4-byte) longwords. Data is displayed in the hex format selected, and also in character format with non-printable characters represented by dots.

3) *editbuf*

   Allows individual buffer locations to be modified in byte, 2-byte or 4-byte units.

4) *fillbuf*

   Allows sections of the buffer to be filled with a repeating pattern. It prompts for start location and length to fill, and for a string of data to use as the fill pattern. (Unfortunately only a string is accepted, it is not possible to enter hex data. The buffer may be cleared by entering a null string).
5) number
Accepts a decimal number, and prints it in octal and hex. For ESDI and SMD drive types only, it also prints the input, interpreted as a blocknumber, in the form cylinder/head/sector for the current drive. It will also accept input in the form cylinder/head/sector (eg 123/4/5) and print the corresponding blocknumber. (SCSI drives are always accessed by a single blocknumber, so this translation is not performed if the current drive type is SCSI).

6) readbuf
This allows disk blocks to be read into the internal buffer. It prompts for buffer address (relative to start of buffer), and number of blocks to read. Up to 100 blocks may be read in one operation. The disk block address from which the read will occur is maintained as an internal variable by \( fx \), it can be set with the seek function.

7) seek
This sets the internal \( fx \) variable which holds the source or destination blocknumber on disk for transfers between disk and the internal buffer. A prompt of the current value is given.

The remaining functions appear only in extended mode, since they are either potentially destructive (eg writebuf) or of little interest to the normal user.

8) writebuf
This writes blocks from the internal buffer to the disk. It prompts for source buffer address and number of blocks to write. The disk address block for the write is taken from the internal \( fx \) variable set by seek, as for readbuf.

9) showuib
This function appears only for ESDI or SMD drives. It prints the Unit Initialization parameters used by the disk controller for the current drive.

10) showstatus
This function appears only for ESDI or SMD drives. It prints the firmware ID code of the Disk controller, and the status of the controller and drive.

11) rawreadbuf
This function appears only for ESDI or SMD drives. It is something of a misnomer, since what it actually does is to invoke the disk controller command intended for reading ESDI or SMD manufacturer’s flawmaps at the start of tracks. It prompts for destination buffer address, disk address and length. It should be noted that the sector part of disk address is irrelevant, since the controller
always reads at start of track for this function. Also, the length parameter is ignored! The controller always transfers one sector for this operation.

12) rawreaddc

This function appears only for Interphase controllers. It is a variant of the previous function, invoking a flawmap read function of the Interphase controller intended to compensate for the different format of flawmaps on CDC disks.

13) passthrough

This appears only because the runtime fx utility shares common code with the standalone version. Direct passthrough of drive command codes is not allowed at runtime.

EXERCISE MENU

This gives access to functions intended for surface analysis of the disk to find bad blocks. Only read-only tests are possible in normal (non-extended) mode; destructive read-write tests are allowed in extended mode.

1) butterfly

Invokes a test pattern in which successive transfers cause seeks to widely separated areas of the disk. This is intended to stress the head positioning system of the drive, and will sometimes find errors which do not show up in a sequential test. It prompts for the range of disk blocks to exercise, number of scans to do, and a test modifier. Each of the available test patterns may be executed in a number of different modes (read-only, read-write etc) which will be described below.

2) errlog

This prints the total number of read and write errors which have been detected during a preceding exercise.

3) random

Invokes a test pattern in which the disk location of successive transfers is selected randomly; intended to simulate a multiuser load. As with the butterfly test, it prompts for range of blocks to exercise, number of scans, and modifier.

4) sequential

Invokes a test pattern in which the disk surface is scanned sequentially. As with the butterfly test, it prompts for range of blocks to exercise, number of scans, and modifier.
The following items appear only in extended mode, since they are concerned with destructive (write) tests.

5) settestpat
   This allows the pattern of data which will be used in tests which write to the disk to be created. One sector of data may be initialized, byte by byte. Each byte may be entered as a decimal or hex value.

6) showtestpat
   This displays the pattern of data which will be used in tests which write to the disk. The default is a repeating pattern of 0xdb 0x6d 0xb6. This may be changed with settestpat.

7) complete
   This causes a write-and-compare sequential test to be run on the entire disk area.

The butterfly, random and sequential tests prompt for a modifier which determines the type of transfer which will occur during the test patterns. Possible modifiers are:

rd-only This causes a read of one track of the disk at each location in the test pattern. The value of read data is ignored, the test detects only the success or failure of the read operation.

rd-cmp This causes two reads of the disk track at each location in the test pattern. The data obtained in the two reads is compared.

seek With this modifier, each sector of each track in the test pattern is read individually in a one-sector-read operation to verify individual sector addressability. (This is a rather time-consuming operation!)

The following modifiers are presented and legal only in extended mode, since they cause writing to the disk, thereby destroying existing data.

wr-only This causes data to be written to one track of the disk at each location in the test pattern. Written data is not re-examined, the test detects only the success or failure of the write operation.

wr-cmp This causes data to be written to one track of the disk at each location in the test pattern. The written data is then read back and compared with its expected value.

**LABEL MENU**
This gives access to functions for displaying, and (in extended mode) modifying information contained in the disk label. It contains the following items:
1) \textit{readin}

Allows part or all of the label to be read in from the disk. Selecting this item brings up a menu of the accessible parts of the label. (These are described in detail below). Selecting a part causes that part to be read in from disk; there is also an \textit{all} option, to read in all parts at once.

2) \textit{show}

Allows display of parts of the label. As for \textit{readin}, it brings up a menu of the label parts, allowing selection of the part to be displayed.

The remaining items appear only in extended mode, since they offer the possibility of changing data on the disk.

3) \textit{sync}

Writes the in-core copy of the disk label back to disk.

4) \textit{set}

Allows parts of the label to be modified. As for \textit{readin}, it brings up a menu of the label parts, allowing selection of the part to be modified.

5) \textit{create}

Discards existing label information, and creates new label information. If the label on disk is valid, the created label information is based on this, otherwise default label information based on the drive type is created. This would normally be used only for attempting to repair a damaged disk label (or to recover from major errors during \textit{set}). As for \textit{readin}, it brings up a menu of the label parts, allowing selection of the part to be worked on.

\textbf{Parts of the disk label}

A disk label contains the following parts:

1) \textit{parameters}

This is information used by the disk controller, such as disk geometry (e.g., number of cylinders), and format information (e.g., interleave). It may be viewed under the \textit{show} submenu of the \textit{label} menu, and in extended mode may be changed by the \textit{set} submenu or reset to default values for the drive type by the \textit{create} submenu. The parameters concerned will depend on the type of controller. These values will not need to be changed in normal use; a full discussion is beyond the scope of this document and the reader should refer to the manufacturer’s documentation for the disk controller and disk drive.

2) \textit{partitions}

The disk surface is divided for convenience into a number of different sections ("partitions") used for various purposes. (See \textit{intro (7m)} for more details). When the operating system is accessing the
disk, its drivers make the connection between the special file name and the physical disk partition using information from the partition table in the disk label. This information may be displayed under the `show` submenu of the `label` menu, and in extended mode changed under the `set` submenu or reset to default values for the drive type by the `create` submenu. There may be up to 16 partitions on a disk, numbered 0 to 15 (though not all need be present). Each partition is described by its starting block on the disk, its size in blocks, and a type indicating its expected use (eg filesystem, disk label, swap etc).

3) `sgiinfo`  
This contains information kept for administrative purposes: the type of disk drive and its serial number. As with other parts of the label, it may be viewed under the `show` submenu of the `label` menu, and in extended mode may be changed by the `set` submenu or reset to default values for the drive type by the submenu.

4) `bootinfo`  
This contains information used by the system proms during a normal system boot. It specifies the root partition, the name of the file on the root partition to boot, and the swap partition. Normal defaults for these are root partition 0, filename `unix` and swap partition 1. Current setting may be viewed under the `show` submenu of the `label` menu, and in extended mode changed under the `set` submenu. The defaults appear as prompts, and may be changed by inputting different values.

5) `directory`  
Some system files are normally kept in the label area on the disk. These are files used in standalone operations such as the standalone shell `sash` and the standalone version of `fx`. The directory is a table in the label which enables these files to be located. The `show` submenu of the `label` menu allows the directory of these files to be displayed.

The files in the disk label are manipulated by the use of `dvhtool (1m)` and `fx` does not provide facilities for adding or deleting them.

**INITIALIZING NEW DISKS**  
`fx` may be used to initialize disk drives which have not been previously formatted.

The new drive to be initialized should be physically connected to the system.

**Warning:** this should not be attempted while the system is powered up!
Take care that termination of the new drive is correct and that its drive id does not conflict with that of any other drive connected to the same controller. With the new drive connected, bring the system back up to normal multiuser mode, and invoke fx in extended mode. Enter the controller type and number, and the drive number for the new drive. For SCSI drives, the drive type will be determined automatically by an inquiry operation on the drive. For ESDI or SMD drives, a menu of known drive types will appear; it is important to know the exact model number of the drive you are adding. (Note that it is possible to work with drives other than those on the menu by entering a drive name of "other", and then entering parameters for the drive. This is intended only for engineering tests and evaluations. Use of drives not qualified by Silicon Graphics Inc is not recommended).

Once drive type is identified, select the auto item on the main menu; this will format the drive, scan it for bad blocks, and place a label on it. On completion of this, exit from fx; the drive is now ready for use. Note that for the new drive to be useful in the system, it will be necessary to make filesystems on it and to mount these filesystems. See mkfs (1M) and mount (1M).

FILES
/dev/rdsk/ips*, /dev/rdsk/dks*, /dev/rdsk/xyl*, /dev/rdsk/fds*

SEE ALSO
xyl(7), dkip(7), dksc(7), smfd(7), dvhtool(1M), vh(7).
NAME

gated – gateway routing daemon

SYNOPSIS

/usr/etc/gated [ -t[ierpuRH] ] [ logfile ]

DESCRIPTION

Gated is a routing daemon that handles multiple routing protocols and
replaces routed(1M), egpup(1M), and any routing daemon that speaks the
HELLO routing protocol. Gated currently handles the RIP, EGP, and
HELLO routing protocols. Gated can be configured to perform all routing
protocols or any combination of the three. The configuration for gated is
stored in the file /usr/etc/gated.conf.

COMMAND LINE TRACING OPTIONS

Gated can be invoked with a number of tracing flags and/or a log file. Trac-
ing flags may also be specified in the configuration file with the
“traceflags” clause. Gated forks and detaches itself from the controlling
terminal unless tracing flags are specified without specifying a log file, in
which case all tracing output is sent to the controlling terminal. The valid
tracing flags are as follows:

-t    If used alone, log all error messages, route changes and EGP pack-
etts sent and received. Using this flag alone turns on the i, e, r, and
p trace flags automatically. When used with another flag, the -t has
no effect and only the accompanying flags are recognized.
    Note that when using other flags, -t must be used with them.

i    Log all internal errors and interior routing errors.

e    Log all external errors due to EGP, exterior routing errors, and
EGP state changes.

r    Log all routing changes.

p    Trace all EGP packets sent and received.

u    When used with p, R, H or N, display the entire contents of routing
packets sent and received.

R    Trace all RIP packets sent or received.

H    Trace all HELLO packets sent or received.

N    Trace all SNMP transactions.

Gated always logs fatal errors. If no log file is specified and no tracing
flags are set, all messages are sent to /dev/null.
SIGNAL PROCESSING

Gated catches a number of signals and performs specific actions. Currently gated does special processing with the SIGHUP, SIGINT and SIGUSR1 signals.

When a SIGHUP is sent to gated and gated is invoked with trace flags and a log file, tracing is toggled off and the log file is closed. At this point the log file may be moved or removed. The next SIGHUP to gated will toggle the tracing on. Gated reads the configuration file and sets the tracing flags to those specified with the "traceflags" clause. If no "traceflags" clause is specified tracing is resumed using the trace flags specified on the command line. The log file specified in the command line is created if necessary and the trace output is sent to that file. The trace output is appended to an already existing log file. This is useful for having rotating log files like those of the syslog(1M) daemon.

Sending gated a SIGINT will cause a memory dump to be scheduled within the next sixty seconds. The memory dump will be written to the file /usr/tmp/gated_dump. Gated will finish processing pending routing updates before performing the memory dump. The memory dump contains a snapshot of the current gated status, including the interface configurations, EGP neighbor status and the routing tables. If the file /usr/tmp/gated_dump already exists, the memory dump will be appended to the existing file.

On receipt of a SIGUSR1, gated will reread selected information from the configuration file. This information currently includes the "announcetoAS", "noannouncetoAS" and "validAS". If no errors are detected the new configuration information is put into effect, if errors are detected, the configuration information is not changed. Gated will also check the interface status on receipt of a SIGUSR1.

CONFIGURATION FILE OPTIONS FOR CONTROLLING TRACING OUTPUT

traceflags traceflag [traceflag] [traceflag] ...

The clause tells the gated process what level of tracing output is desired. This option is read during gated initialization and whenever gated receives a SIGHUP. This option is overridden at initialization time if tracing flags are specified on the command line. The valid tracing flags are as follows:

internal Log all internal errors and interior routing errors.

external Log all external errors due to EGP, exterior routing errors, and EGP state changes.

route Log all routing changes.
egp
Trace all EGP packets sent and received.

update
When used with egp, rip, hello or snmp, display the contents of all routing packets sent and received.

rip
Trace all RIP packets sent and received.

hello
Trace all HELLO packets sent and received.

icmp
Trace all ICMP redirect packets received.

snmp
Trace all SNMP transactions.

stamp
Print a timestamp to the log file every 10 minutes.

general
A combination of "internal", "external", "route" and "egp".

all
Enable all of the above tracing flags.

If more than one "traceflags" clause is used, the tracing flags accumulate.

DEFAULT CONFIGURATION
Gated normally reads configuration information from /usr/etc/gated.conf. If this file does not exist, gated assumes a default configuration file of:

    RIP yes
    HELLO no
    EGP no

In addition, if the configuration file does not exist, there is only one network interface, and a default route is installed in the kernel, gated will exit assuming that a simple default route is adequate.

CONFIGURATION FILE OPTIONS FOR HANDLING ROUTING PROTOCOLS
In this section, the numerous configuration options are explained. Each time the gated process is started, it reads the file /usr/etc/gated.conf to obtain its instructions on how routing will be managed with respect to each protocol. The configuration options are as follows:

RIP {yes | no | supplier | pointpoint | quiet | gateway #}

This tells the gated process how to perform the RIP routing protocol. Only one of the above RIP arguments is allowed after the keyword "RIP". If more than one is specified, only the first one is recognized. A list of the arguments to the RIP clause follows:

yes
Perform the RIP protocol. Process all incoming RIP packets and supply RIP information every thirty seconds only if there are two or more network interfaces.
no

Do not perform the RIP protocol. Do not perform RIP.

supplier

Perform the RIP protocol. Process all incoming RIP packets and force the supplying of RIP information every thirty seconds no matter how many network interfaces are present.

pointopoint

Perform the RIP protocol. Process all incoming RIP packets and force the supplying of RIP information every thirty seconds no matter how many network interfaces are present. When this argument is specified, RIP information will not be sent out in a broadcast packet. The RIP information will be sent directly to the gateways listed in the "sourceripgateways" option described below.

quiet

Process all incoming RIP packets, but do not supply any RIP information no matter how many network interfaces are present.

gateway #

Process all incoming RIP packets, supply RIP information every thirty seconds, and announce the default route (0.0.0.0) with a metric of #. The metric should be specified in a value that represents a RIP hopcount. With this option set, all other default routes coming from other RIP gateways will be ignored. The default route is only announced when actively peering with at least one EGP neighbor and therefore should only be used when EGP is used.

If no "RIP" clause is specified, RIP will not be performed.

HELLO {yes | no | supplier | pointopoint | quiet| gateway #}

This tells gated how to perform the HELLO routing protocol. The arguments parallel the RIP arguments, but do have some minor differences. Only one of the above HELLO arguments is allowed after the keyword "HELLO". If more than one is specified, only the first one is recognized. A list of the arguments to the HELLO clause follows:

yes

Perform the HELLO protocol. Process all incoming HELLO packets and supply HELLO information every fifteen seconds only if there are two or more network interfaces.

no

Do not perform the HELLO protocol. Do not perform HELLO.

supplier

Perform the HELLO protocol. Process all incoming HELLO packets and force the supplying of HELLO information every fifteen seconds no matter how many network interfaces are present.
pointtopoint  Perform the HELLO protocol. Process all incoming HELLO packets and force the supplying of HELLO information every fifteen seconds no matter how many network interfaces are present. When this argument is specified, HELLO information will not be sent out in a broadcast packet. The HELLO information will be sent directly to the gateways listed in the ‘‘sourcehellogateways’’ option described below.

quiet  Process all incoming HELLO packets, but do not supply any HELLO information despite the number of network interfaces present.

gateway #  Process all incoming HELLO packets, supply HELLO information every fifteen seconds, and announce the default route (0.0.0.0) with a time delay of #. The time delay should be specified in milliseconds. The default route is only announced when actively peering with at least one of EGP neighbor, therefore should only be used when running EGP.

If no ‘‘HELLO’’ clause is specified, HELLO will not be performed.

EGP {yes | no}  
This clause allows the processing of EGP by gated to be turned on or off.

no  Do not perform any EGP processing.

yes  Perform all EGP operations.

Please note that by default, EGP processing will take place. Therefore, if no ‘‘EGP’’ clause is specified, all EGP operations will take place.

autonomoussystem #  
If performing the EGP protocol, this clause must be used to specify the autonomous system number (#). If not specified, gated will exit and give a fatal error message.

egpmaxacquire #  
If performing the EGP protocol, this clause specifies the number of EGP peers with whom gated will be performing EGP. This number must be greater than zero and less than or equal to the number of EGP neighbors specified or gated will exit. If this clause is omitted, all EGP neighbors will be acquired.
egpneighbor  gateway
metricin metric
egpmetricout egpmetric
ASin asin
ASOut asout
AS as
nogendefault
acceptdefault
defaultout egpmetric
validate
intf interface
 sourcenet net
gateway gateway

If performing the EGP protocol, this clause specifies with whom gated will be performing EGP. "Gateway" can be either a symbolic name in /etc/hosts or an IP hostname in Internet dot (a.b.c.d) notation. Dot notation is recommended to avoid confusion. Each EGP neighbor will be acquired in the order listed in the configuration file.

The "metricin" option is used to specify the internal time delay to be used as a metric for all of the routes learned from this neighbor. It should be specified as a time delay from zero to 30000. If this option and the validate option are not used, the internal metric used is the EGP distance multiplied by 100.

The "egpmetricout" option is used to specify the EGP distance used for all nets advertised to this neighbor. It should be specified as an EGP distance in the range of 0 to 255. If this option is not specified, the internal time delay for each route will be converted to an EGP distance by division by 100 with distances greater than 255 being set to 255.

The "ASin" option is used to verify the autonomous system number of this neighbor. If the autonomous system number specified in neighbor acquisition packets does not verify an error message is generated refusing the connection. If this option is not specified, no verification of autonomous system numbers is done.

The "ASOut" option is used to specify the autonomous system number in EGP packets sent to this neighbor. If not specified, the autonomous system specified in the "autonomousssystem" clause is used. This clause should not normally be used, it is reserved for a special situation interfacing between the ARPA net and NSFnet.
The "AS" option is used to specify that autonomous system number that will be assigned to routes learned from this neighbor. If not specified, the autonomous system used in the EGP packets received from this neighbor will be used. This clause should not normally be used, it is reserved for a special situation interfacing between the ARPAnet and NSFnet.

The "nogendefault" option is used to specify this neighbor should not be considered for the internal generation of default when "RIP gateway" or "HELLO gateway" is used. If not specified, the internal default will be generated when actively peering with this neighbor.

The "acceptdefault" option is used to specify that the default route (net 0.0.0.0) should be considered valid when received from this neighbor. If this option is not specified, the reception of the default route will cause a warning message to be printed and the route to be ignored.

The "defaultout" option is used to specify that the internally generated default may be passed to this EGP neighbor at the specified distance. The distance should be specified as an EGP distance from 0 to 255. A default route learned from another gateway will not be propagated to an EGP neighbor. Normally, no default route will be passed via EGP. The "acceptdefault" option should not be specified when the "defaultout" option is used. The egpnmetric specified in the "egpnmetricout" option does not apply, the default route will always use the metric specified by the "defaultout" option.

The "validate" option is used to specify that all networks received from this EGP neighbor must be specified in "validAS" clause that also specifies the autonomous system of this neighbor. Networks not having a "validAS" clause will be ignored after a warning message is printed.

The "intf" option is used to specify the interface used to send EGP packets to this neighbor. This option is only required when there is no common net/subnet with this egpnneighbor. This option currently is only present for testing purposes and does not imply correct operation when peering with an egpnneighbor that does not share a common net/subnet.

The "sourcenet" option is used to specify the source network to be specified in EGP poll packets sent to this neighbor. If this option is not specified, the network (not subnet) of the interface used to communicate with this neighbor is used. This option is currently only present for testing purposes and does not imply correct operation when used.

The "gateway" option is used to specify the gateway to be used when installing routes learned from an EGP neighbor on a different network. Normally these routes would be ignored. This option is currently only present for testing purposes and correct operation can not be assured when it is used.
CONFIGURATION FILE OPTIONS FOR HANDLING ROUTING INFORMATION

The following configuration file options tell gated how to deal with both incoming and outgoing routing information.

trustedripgateways gateway [gateway] [gateway] .....  
trustedhellogateways gateway [gateway] [gateway] .....  

When these clauses are specified, gated will only listen to RIP or HELLO information, respectively from these RIP or HELLO gateways. ‘‘gateway’’ can be either a symbolic name from /etc/hosts or an IP host address in dot notation (a.b.c.d). Again, dot notation is recommended to eliminate confusion. Please note that the propagation of routing information is not restricted by this clause.

sourceripgateways gateway [gateway] [gateway] .....  
sourcehellogateways gateway [gateway] [gateway] .....  

Gated will send RIP or HELLO information directly to the gateways specified. If “pointtopoint” is specified in the “RIP” or “HELLO” clauses mentioned above, gated will only send RIP or HELLO information to the specified gateways. Gated will NOT send out any information using the broadcast address. If “pointtopoint” is not specified in those clauses and gated is supplying of RIP or HELLO information, gated will send information to the specified gateways as well as broadcasting it using a broadcast address.

noripoutinterface intfaddr [intfaddr] [intfaddr] .....  
nohellocoutinterface intfaddr [intfaddr] [intfaddr] .....  
noripfrominterface intfaddr [intfaddr] [intfaddr] .....  
nohellocfrominterface intfaddr [intfaddr] [intfaddr] .....  

The above clauses turn protocols on and off on a per interface basis. “no(rip|hello)frominterface” means that no RIP or HELLO information will be accepted coming into the listed interfaces from another gateway. “no(rip|hello)outinterface” means that no RIP or HELLO knowledge will be sent out of the listed interfaces. “intfaddr” should be in dot notation (a.b.c.d).

passiveinterfaces intfaddr [intfaddr] [intfaddr] .....  

In order to dynamically determine if an interface is properly functioning, gated will time out an interface when no RIP, HELLO or EGP packets are being received on that particular interface. PSN interfaces send a RIP or HELLO packet to themselves to determine if the interface is properly functioning as the delay between EGP packets may be longer than the interface timeout. Interfaces that have timed out automatically have their routes re-installed when routing information is again received over the interface. The
above clause stops gated from timing out the listed interfaces. The interfaces listed will always be considered up and working. If gated is not a RIP or HELLO supplier, all interfaces will not be aged and the "passiveinterfaces" automatically applies to all interfaces.

interface metric\texttt{intfaddr metric#}

This feature allows the specification of an interface metric for the listed interface. On systems that support interface metrics, this clause will override the kernel's metric. On systems that do not have support for an interface metric, this feature allows the specification of one. The interface metric is added to the true metric of each route that comes in via routing information from the listed interface. The interface metric is also added to the true metric of any information sent out via the listed interface. The metric of directly attached interfaces is also set to the interface metric, routing information broadcast about directly attached nets will be based on the interface metric specified. This clause is required for each interface on which an interface metric is desired.

reconst metric\texttt{intfaddr metric#}

This is a first attempt to throw hooks for fallback routing into gated. If the above clause is used, the metrics of the routes contained in any RIP information coming into the listed interface will be set to the specified "metric#". Metric reconstitution should not be used lightly, since it could be a major contributor in the formation of routing loops. \textbf{USE THIS WITH EXTREME CAUTION.} Any route that has a metric of infinity will not be reconstituted and left as infinity.

fixedmetric\texttt{intfaddr proto \{rip\|hello\} metric#}

This is another attempt to throw hooks for fallback routing into gated. If the above clause is used, all routing information sent out the specified interface will have a metric of "metric#". For RIP, specify the metric as a RIP hopcount from 0 to infinity. For HELLO, specify the metric as a HELLO delay in milliseconds from 0 to infinity. Any route that has a metric of infinity will be left as infinity. Fixed metrics should also be \textbf{USED WITH EXTREME CAUTION!}

donotlisten\texttt{net intf addr [addr] ... proto \{rip\|hello\}}
donotlistenhost\texttt{host intf addr [addr] ... proto \{rip\|hello\}}

This clause reads as follows: keyword "donotlisten" followed by a network number, which should be in dot notation followed by keyword "intf". Then a list of interfaces in dot notation precede the keyword "proto", followed by "rip" or "hello".
This means that any information regarding "net" coming in via the specified protocols AND from the specified interfaces will be ignored. The keyword "all" may be used after the keyword "intf" to specify all interfaces on the machine. For example:

`donotlisten 10.0.0.0 intf 128.84.253.200 proto rip`

means that any RIP information about net 10.0.0.0 coming in via interface 128.84.253.200 will be ignored. One clause is required for each net on which this restriction is desired.

`donotlisten 26.0.0.0 intf all proto rip hello`

means that any RIP and HELLO information about net 26.0.0.0 coming in via any interface will be ignored.

"donotlistencodec" can be described the same way as above except that a host address is provided instead of a network address. Restrictions of the nature described above are applied to the specified host route learned of by the specified routing protocol.

`listen net gateway addr [addr] ... proto {rip|hello}`

`listenhost host gateway addr [addr] ... proto {rip|hello}`

This clause reads as follows: keyword "listen" followed by a network number which should be in dot notation followed by keyword "gateway". Then a list of gateways in dot notation should precede the keyword "proto", followed by "rip" or "hello".

This means to only listen to information about network "net" by the specified protocol(s) only from the listed "gateways". For example:

`listen 128.84.0.0 gateway 128.84.253.3 proto hello`

means that any HELLO information about net 128.84 coming in via gateway 128.84.253.3 will be accepted. Any other information about 128.84 from any other gateway will be rejected. One clause is necessary for each net to be restricted.

`listenhost 26.0.0.15 gateway 128.84.253.3 proto rip`

means that any information about host 26.0.0.15 must come via RIP and from gateway 128.84.253.3. All other information regarding this host will be ignored.

`announce net intf addr [addr] ... proto type [egpmetric #]`

`announcehost host intf addr ... proto type [egpmetric #]`

`noannounce net intf addr [addr] ... proto type [egpmetric #]`

`noannouncehost host intf addr ... proto type [egpmetric #]`
These clauses allow restriction of the networks and hosts announced and by which protocol. The "announce{host}" and "noannounce{host}" clauses may not be used together on the same interface. With the "announce{host}" clause, gated will only announce the nets or hosts that have an associated "announce{host}" clause with the appropriate protocol. With the "noannounce{host}" clause, gated will announce everything, EXCEPT those nets or hosts that have an associated "noannounce{host}" clause. This allows a choice of announcing only what is on the announce list or everything except those nets on the noannounce list on a per interface basis.

The arguments are the same as in the "donotlisten" clause except "egp" may be specified in the "proto" field. "type" can either be "rip", "hello", "egp", or any combination of the three. When "egp" is specified in the "proto" field, an egp metric must be specified. This is the metric at which gated will announce the listed net via EGP.

Please note that these are not static route entries. These restrictions will only apply if the net or host is learned via one of the routing protocols. If a restricted network suddenly becomes unreachable and goes away, announcement of this net will stop until it is learned again.

Currently, only one "announce{host}" or "noannounce{host}" may be specified per network or host. It is not possible to announce a network or host via HELLO out one interface and via RIP out another.

Some examples:

```
announce 128.84 intf all proto rip hello egp egpmetric 0
announce 10.0.0.0 intf all proto rip
announce 0.0.0.0 intf 128.84.253.200 proto rip
announce 35.0.0.0 intf all proto rip egp egpmetric 3
```

With only these four "announce" clauses in the configuration file, gated will only announce these four nets. It will announce 128.84.0.0 via RIP and HELLO to all interfaces and announce it via EGP with a metric of 0. Net 10.0.0.0 will be announced via RIP to all interfaces. Net 0.0.0.0 (default) will be announced by RIP out interface 128.84.253.200 only. Net 35.0.0.0 will be announced via RIP to all interfaces and announced via EGP with a metric of 3. These are the only nets that will be broadcast by this gateway. Once the first "announce" clause is specified, only the nets with "announce" clauses will be broadcast; this includes local subnets. Once an "announce{host}" or "noannounce{host}" has an "all" specified after an "intf", that clause is applied globally and the option of having per interface restrictions is lost. If no routing announcement restrictions are desired, "announce" clauses should not be used. All information learned will then be propagated out. Please note that this has no affect on the information to
which *gated* listens. Any net that does not have an "announce" clause is still added to the kernel routing tables, but it is not announced via any of the routing protocols. To stop nets from being added to the kernel the "donotlisten" clause may be used.

```
announce 128.84 intf 128.59.2.1 proto rip
noannounce 128.84 intf 128.59.1.1 proto rip
```

The above clauses mean that on interface 128.59.2.1, only information about 128.84.0.0 will be announced via RIP, but on interface 128.59.1.1, all information will be announced, except 128.84.0.0 via RIP.

```
noannounce 128.84 intf all proto rip hello egp egpmetric 0
noannounce 10.0.0.0 intf all proto hello
```

These clauses mean that except for the two specified nets, all nets will be propagated. Specifically, net 128.84.0.0 will not be announced on any interface via any protocols. Knowledge of 128.84.0.0 is not sent anywhere. Net 10.0.0.0 will not be announced via HELLO to any interface. This also implies that net 10.0.0.0 will be announced to every interface via RIP. This net will also be broadcast via EGP with a metric specified in the "defaultegpmetric" clause.

**defaultegpmetric #**

This is a default EGP metric to use when there are no routing restrictions. Normally, with no routing restrictions, *gated* announces all networks learned via HELLO or RIP via EGP with this specified default EGP metric. If this clause is not used, the default EGP metric is set to 255, which would make any EGP advertised route of this nature be ignored. When there are no routing restrictions, any network with a direct interface is announced via EGP with a metric of 0. Note that this does not include subnets, but only the non-subnetted network.

**defaultgateway gateway proto [metric metric] {active|passive}**

This default gateway is installed in the kernel routing tables during initialization and reinstalled whenever information about the default route is lost. This route is installed with the time delay equivalent of a RIP metric of 15 unless another metric is specified with the metric option.

If 'RIP gateway' or 'HELLO gateway' are in use this default route is deleted when successfully peering with an EGP neighbor not specified for "nogendefault".

An "active" default route will be overridden by any other default route learned via another routing protocol. A "passive" default route will only be overridden by a default route with a lower metric.
An "active" default route will not be propagated in routing updates, a "passive" default route will be propagated.

"gateway" should be an address in dot notation. "metric" is optional and should be a metric in the specified protocol between zero and infinity, if not specified a RIP metric of 15 is used. "proto" should be either "rip", "egp", or "hello". The "proto" field initializes the protocol by which the route was learned. Although in this case it is unused, but the field is remains for consistency.

net netaddr gateway addr metric hopcnt {rip|egp|hello}
host hostaddr gateway addr metric hopcnt {rip|egp|hello}

The following clauses install a static route to net "netaddr" or host "hostaddr" through gateway "addr" at a metric of "hopcnt" learned via either RIP, HELLO, or EGP. As usual, dot notation is recommended for the addresses. This route will be installed in the kernel's routing table and will never be affected by any other gateway's RIP or HELLO announcements. The protocol by which it was learned is important if the route is to be announced via EGP. If the protocol is "rip" or "hello" and there are no routing restrictions, then this route will be announced by EGP with a metric of "defaultegpmetric". If the protocol is "egp" and there are no routing restrictions, then this route will be announced by EGP with a metric of "hopcnt".

egpnetsreachable net [net] [net] ..... 

This option was left in as a "soft restriction". It cannot be used when the "announce" or "noannounce" clauses are used. Normally, with no restrictions, gated announces all routes learned from RIP and HELLO via EGP. The "egpnetsreachable" clause restricts EGP announcement to those nets listed in the clause. The metric used for the HELLO and RIP learned routes is the value given in the "defaultegpmetric" clause. If this clause does not specify a value, the value is set to 255. With the "egpnetsreachable" clause, individual unique EGP metrics may not be set for each net. The "defaultegpmetric" is used for all networks except those that are directly connected, which use a metric of 0.

martiannets net [net] [net] ...

This clause appends to gated's list of "martian" networks. "Martian" networks are those known to be invalid and should be ignored. When gated hears about one of these networks through any means, it will immediately ignore it. If "external" tracing is enabled, a message will be printed to the trace log. Multiple occurrences of the "martiannets" clause accumulate.
An initial list of "martian" networks is coded into gated. This list contains 127.0.0.0, 128.0.0.0, 191.253.0.0, 192.0.0.0, 223.255.255.0, and 224.0.0.0.

CONFIGURATION FILE OPTIONS FOR AUTONOMOUS SYSTEM ROUTING

In the internal routing tables, gated maintains the autonomous system number from which each route was learned. Autonomous systems are used only when an exterior routing protocol is in use, in this case EGP. Routes are tagged with the autonomous system number of the EGP peer from which they were learned. Routes learned via the interior routing protocols, RIP and HELLO, are tagged with the autonomous system number specified in the "autonomous system" clause.

Gated normally does not propagate routes learned from exterior routing protocols to interior routing protocols. Historically this is because of the ARPANET core EGP speakers which do not have adequate validation of routing information they receive. Some of the following clauses allow exterior routes to be propagated via interior protocols. Therefore it is imperative that utmost care be taken when allowing the propagation of exterior routes.

The following clauses provide limited control over routing based on autonomous system number.

validAS net AS as metric metric

The "validAS" clause is used for validation of networks from certain AS. When an EGP update is received from a neighbor which has the "validate" option specified on the associated "egpneighbor" clause a "validAS" clause is searched for specifying that network and the autonomous system number of the EGP neighbor. If the appropriate "validAS" clause is located, the network is considered for addition to the routing table with the specified metric. If a "validAS" clause is not located, a warning message is printed and the network is ignored.

A network may be specified in several "validAS" clauses as being associated with several different autonomous systems.

announcetoAS as0 {restrict|norestrict} ASlist as1 as2 as3 ...
noannouncetoAS as0 {restrict|norestrict} ASlist as1 as2 as3 ...

The "announcetoAS" and "noannouncetoAS" control the exchanging of routing information between different autonomous systems. Normally gated will not propagate routing information between autonomous systems. The exception to this is that routes learned from gated's own autonomous system via RIP and HELLO will be propagated via EGP. These clauses allow information learned via EGP from one autonomous system to be propagated via EGP to another autonomous system or via RIP and HELLO to gated's own autonomous system.
If the "announce-to-AS" clause is specified, information learned via EGP from autonomous systems \( as1, as2, as3, \ldots \) will be propagated to autonomous system \( as0 \). If \( gated \)'s own autonomous system, as specified in the "autonomous-system" clause, is specified as \( as0 \), this information will be propagated via RIP and HELLO. Routing information from autonomous systems not specified in the ASlist will not be propagated to autonomous system \( as0 \).

If the "no-announce-to-AS" clause is specified, information learned via EGP from all autonomous systems except \( as1, as2, as3, \ldots \) will be propagated to autonomous systems \( as0 \). If \( gated \)'s own autonomous system is specified as \( as0 \), this information will not be propagated via RIP and HELLO.

The "[no]restrict" option controls the application of "announce" and "no-announce" clauses to the propagation of routes to different autonomous systems. If "restrict" is specified, normal announcement restrictions do apply, if "norestrict" is specified, announcement restrictions are not considered, all routes from the source autonomous systems are propagated to the destination autonomous system.

Only one "announce-to-AS" or "no-announce-to-AS" clause may be specified per target autonomous system.

**NOTES ON CONFIGURATION OPTIONS**

If EGP is being used when supplying the default route (via "RIP gateway" or "HELLO gateway") and all EGP neighbors are lost, the default route will not be advertised until at least one EGP neighbor is regained.

With the complexity of the current network topology and with many backdoor paths to networks, the use of routing restrictions is recommended. With the current routing strategies, it is easy for illegal or invalid networks to penetrate into the ARPAnet Core or the NSNet backbone. Using routing restrictions does take a little more maintenance time and routing restrictions are not the long term answer, but, for now, in order to be good Internet players, we must use them.

**GATED INTERNAL METRICS**

\( Gated \) stores all metrics internally as a time delay in milliseconds to preserve the granularity of HELLO time delays. The internal delay ranges from 0 to 30000 milliseconds with 30000 representing infinity. Metrics from other protocols are translated to and from a time delay as they are received and transmitted. EGP distances are not comparable to HELLO and RIP metrics but are stored as a time delay internally for comparison with other EGP metrics. The conversion factor between EGP distances and time delays is 100. RIP and interface metrics are translated to and from the internal time delays with the use of the following translation tables:
<table>
<thead>
<tr>
<th>Time Delay</th>
<th>RIP metric</th>
<th>RIP metric</th>
<th>Time Delay</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>100</td>
<td>1</td>
<td>100</td>
</tr>
<tr>
<td>101</td>
<td>148</td>
<td>2</td>
<td>148</td>
</tr>
<tr>
<td>149</td>
<td>219</td>
<td>3</td>
<td>219</td>
</tr>
<tr>
<td>220</td>
<td>325</td>
<td>4</td>
<td>325</td>
</tr>
<tr>
<td>326</td>
<td>481</td>
<td>5</td>
<td>481</td>
</tr>
<tr>
<td>482</td>
<td>713</td>
<td>6</td>
<td>713</td>
</tr>
<tr>
<td>714</td>
<td>1057</td>
<td>7</td>
<td>1057</td>
</tr>
<tr>
<td>1058</td>
<td>1567</td>
<td>8</td>
<td>1567</td>
</tr>
<tr>
<td>1568</td>
<td>2322</td>
<td>9</td>
<td>2322</td>
</tr>
<tr>
<td>2323</td>
<td>3440</td>
<td>10</td>
<td>3440</td>
</tr>
<tr>
<td>3441</td>
<td>5097</td>
<td>11</td>
<td>5097</td>
</tr>
<tr>
<td>5098</td>
<td>7552</td>
<td>12</td>
<td>7552</td>
</tr>
<tr>
<td>7553</td>
<td>11190</td>
<td>13</td>
<td>11190</td>
</tr>
<tr>
<td>11191</td>
<td>16579</td>
<td>14</td>
<td>16579</td>
</tr>
<tr>
<td>16580</td>
<td>24564</td>
<td>15</td>
<td>24564</td>
</tr>
<tr>
<td>24565</td>
<td>30000</td>
<td>16</td>
<td>30000</td>
</tr>
</tbody>
</table>

**NOTES ON IMPLEMENTATION SPECIFICS**

In the *gated* configuration file, all references to POINT-TO-POINT interfaces must use the DESTINATION address.

All protocols have a two minute hold down. When a routing update indicates that the route in use is being deleted, *gated* will not delete the route for two minutes.

Changes can be made to the interfaces and *gated* will notice them without having to restart the process. If the netmask, subnetmask, broadcast address, or interface metric are changed, the interface should be marked down with *ifconfig* (1M), then marked up at least thirty seconds later. Flag changes do not require the interface to be brought down and back up.

RIP propagates and listens to host routes, thus allowing the consistent handling of PTP links. The RIP_TRACE commands are also supported.

Subnet interfaces are supported. Subnet information will only be propagated on interfaces to other subnets of the same network. For example, if there is a gateway between two class B networks, the subnet routes for each respective class B net are not propagated into the other class B net. Just the class B network number is propagated.

*Gated* listens to host and network REDIRECTs and tries to take an action on the REDIRECT for its own internal tables that parallels the kernel’s action. In this way, the redirect routine in *gated* parallels the Berkeley kernel redirect routine as closely as possible. Unlike the Berkeley kernel, *gated* times out routes learned via a REDIRECT after six minutes. The
route is then deleted from the kernel routing tables. This helps keep the routing tables more consistent. Any route that was learned via a REDIRECT is NOT announced by any routing protocol.

The gated EGP code verifies that all nets sent and received are valid class A, B or C networks per the EGP specification. Information about networks that do not meet these criteria is not propagated. If an EGP update packet contains information about a network that is not either class A, B or C, the update is considered to be in error and is ignored.

FILES
/usr/etc/gated.conf configuration file.
/usr/tmp/gated_dump memory dump file
/usr/etc/gated gated itself

SEE ALSO
routed(1M)
RFC827 EXTERIOR GATEWAY PROTOCOL (EGP)
RFC888 "STUB" EXTERIOR GATEWAY PROTOCOL
RFC891 DCN Local-Network Protocols (HELLO)
RFC904 Exterior Gateway Protocol Formal Specification
RFC911 EGP GATEWAY UNDER BERKELEY UNIX 4.2

CREDITS
This program was derived from Paul Kirton’s EGP for UNIX, UC at Berkeley’s routed(1M), and HELLO routines by Mike Petry at the University of Maryland.
NAME
getty – set terminal type, modes, speed, and line discipline

SYNOPSIS
/etc/getty [ -h ] [ -t timeout ] [ -s altline ] line [ speed [ type [ linedisc ] ] ]
/etc/getty -c file

DESCRIPTION
getty is a program that is invoked by init(1M). It is the second process in
the series, (init-getty-login-shell) that ultimately connects a user with the
UNIX system. It can only be executed by the super-user; that is, a process
with the user-ID of root. Initially getty prints the contents of /etc/issue (if it
exists), then prints the login message field for the entry it is using from
/etc/gettydefs, reads the user’s login name, and invokes the login(1) com-
mand with the user’s name as argument. While reading the name, getty
attempts to adapt the system to the speed and type of terminal being used.
It does this by using the options and arguments specified.

Line is the name of a tty line in /dev to which getty is to attach itself. getty
uses this string as the name of a file in the /dev directory to open for reading
and writing. Unless getty is invoked with the -h flag, getty will force a
hangup on the line by setting the speed to zero before setting the speed to
the default or specified speed. The -t flag plus timeout (in seconds),
specifies that getty should exit if the open on the line succeeds and no one
types anything in the specified number of seconds.

The -s option specifies the name of a character device special file in the
/dev directory. When this option is specified, getty compares the major and
minor numbers of altline with those of the device special file specified by
line. If these numbers both match, then the two device files represent the
same system device and in this case getty will simply go to sleep and do
nothing. For example, the command

    getty -s console ttys1

causes getty to compare the major and minor numbers of /dev/console and
/dev/ttys1. If the two agree, getty quietly goes to sleep and does nothing.
This allows there to be two lines in the init(4) file for what appear to be
distinct devices even though it may happen that in some configurations
these two devices are actually linked together. When they happen to be the
same, only one of the getty processes started by init(1M) will actually
interact with the device: the one started without the -s option.

Speed, the optional second argument, is a label to a speed and tty definition
in the file /etc/gettydefs. This definition tells getty at what speed to initially
run, what the login message should look like, what the initial tty settings
are, and what speed to try next should the user indicate that the speed is
inappropriate (by typing a `<break>` character).

*Type*, the optional third argument, is a character string describing to *getty* what type of terminal is connected to the line in question. *getty* recognizes the following types:

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td>default</td>
</tr>
<tr>
<td>ds40-1</td>
<td>Dataspeed40/1</td>
</tr>
<tr>
<td>tektronix,tek</td>
<td>Tektronix</td>
</tr>
<tr>
<td>vt61</td>
<td>DEC vt61</td>
</tr>
<tr>
<td>vt100</td>
<td>DEC vt100</td>
</tr>
<tr>
<td>hp45</td>
<td>Hewlett-Packard 45</td>
</tr>
<tr>
<td>c100</td>
<td>Concept 100</td>
</tr>
</tbody>
</table>

The default terminal is *none*; i.e., any crt or normal terminal unknown to the system. Also, for terminal type to have any meaning, the virtual terminal handlers must be compiled into the operating system. They are available, but not compiled in the default condition.

*Linedisc*, the optional fourth argument, is a character string describing which line discipline to use in communicating with the terminal. There are two line disciplines. LDISC0 is the familiar System V line discipline. LDISC1 is similar to the 4.3BSD "new tty driver" (see *termio*(7)). LDISC1 is the default.

When given no optional arguments, *getty* sets the *speed* of the interface to 9600 baud, specifies that raw mode is to be used (awaken on every character), that echo is to be suppressed, either parity allowed, new-line characters will be converted to carriage return-line feed, and tab expansion performed on the standard output. It types the login message before reading the user's name a character at a time. If a null character (or framing error) is received, it is assumed to be the result of the user pushing the "break" key. This will cause *getty* to attempt the next *speed* in the series. The series that *getty* tries is determined by what it finds in `/etc/gettydefs`.

After the user's name has been typed in, it is terminated by a new-line or carriage-return character. The latter results in the system being set to treat carriage returns appropriately (see *ioctl*(2)).

The user's name is scanned to see if it contains any lower-case alphabetic characters; if not, and if the name is non-empty, the system is told to map any future upper-case characters into the corresponding lower-case characters.

Finally, *login* is exec'd with the user's name as an argument. Additional arguments may be typed after the login name. These are passed to *login*, which will place them in the environment (see *login*(1)).
If `getty` is running on the graphics console, `getty` checks to see if autologin is enabled by verifying the existence of `/etc/autologin` and `/etc/autologin.on` (see `login(1)`). If autologin is enabled, `getty` will invoke `login` with the autologin option.

A check option is provided. When `getty` is invoked with the `-c` option and `file`, it scans the file as if it were scanning `/etc/gettydefs` and prints out the results to the standard output. If there are any unrecognized modes or improperly constructed entries, it reports these. If the entries are correct, it prints out the values of the various flags. See `ioctl(2)` to interpret the values. Note that some values are added to the flags automatically.

**FILES**

/etc/gettydefs
/etc/issue message printed before `login` prompt
/etc/autologin
/etc/autologin.on

**SEE ALSO**

`ct(1C)`, `init(1M)`, `tty(7)`.
`login(1)` in the *User’s Reference Manual*.
`ioctl(2)`, `gettydefs(4)`, `inittab(4)` in the *Programmer’s Reference Manual*.

**BUGS**

While `getty` understands simple single character quoting conventions, it is not possible to quote certain special control characters used by `getty`. Thus, you cannot login via `getty` and type a `#`, `@`, `/`, `!`, `_`, `backspace`, `^U`, `^D`, or `&` as part of your login name or arguments. `getty` uses them to determine when the end of the line has been reached, which protocol is being used, and what the erase character is. They will always be interpreted as having their special meaning.
NAME
growfs – expand a filesystem

SYNOPSIS
/etc/growfs [-s size] special

DESCRIPTION
Growfs expands an existing Extent Filesystem, see fs(4). The special argument should be the pathname of the device special file where the filesystem resides. The filesystem must be unmounted to be grown, see umount(1M). The existing contents of the filesystem are undisturbed, and the added space becomes available for additional file storage.

If a size argument is given, the filesystem will be grown to occupy size basic blocks of storage (if available).

If no size argument is given, the filesystem will be grown to occupy all the space available on the device.

It is anticipated that growfs will most often be used in conjunction with logical volumes see lv(7M). However, it can also be used on a regular disk partition, for example if a partition has been enlarged while retaining the same starting block.

PRACTICAL USE
Filesystems normally occupy all of the space on the device where they reside. In order to grow a filesystem, it is necessary to provide added space for it to occupy. Therefore there must be at least one spare new disk partition available.

Adding the space is done through the mechanism of logical volumes.

If the filesystem already resides on a logical volume, the volume is simply extended using mklv(1M).

If the filesystem is currently on a regular partition, it is necessary to create a new logical volume whose first member is the existing partition, with subsequent members being the new partition(s) to be added. Again, mklv is used for this.

In either case growfs is run on the logical volume device, and the expanded filesystem is then available for use on the logical volume device.
DIAGNOSTICS

Growfs will expand only clean filesystems. If any problem is detected with the existing filesystem, the following error message is printed:

"growfs: filesystem on <special> needs cleaning."

If a size argument is given, growfs checks that the specified amount of space is available on the device. If not, it prints the error message:

"growfs: cannot access <size> blocks on <special>..

Growfs works in units of the cylinder group size in the existing filesystem. To usefully expand the filesystem there must be space for at least one new cylinder group. Failing this, it prints the error message:

"growfs: not enough space to expand filesystem."

COMPATIBILITY NOTE

Growfs can expand a filesystem from any IRIX release, and filesystems may be expanded repeatedly. However, once a filesystem has been grown, it is NOT possible to mount it on an IRIX system earlier than release 3.3, and a pre-3.3 fsck will not recognize it.

SEE ALSO

mkfs(1M), mklv(1M), lv(7M).
NAME
halt – halt the system

SYNOPSIS

cd /
/etc/halt

DESCRIPTION

Halt is executed by the super-user to shut the machine down so it is safe to
remove power. halt leaves the machine executing the firmware monitor. If
you are remotely logged-in to the system, you will be prompted to confirm
the shutdown.

SEE ALSO

reboot(1M), shutdown(1M), init(1M).
NAME
hinv – hardware inventory command

SYNOPSIS
hinv [ -v ] [ -s ] [ -c class ] [ -t type ]

DESCRIPTION
hinv displays the contents of the system hardware inventory table. This table is created each time the system is booted and contains entries describing various pieces of hardware in the system. The items in the table include main memory size, cache sizes, floating point unit, and disk drives. Without arguments, the hinv command will display a one line description of each entry in the table. The -v option will give a more verbose description of some items in the table. The -c class option will display items from class. Classes are processor, disk, memory, serial, parallel, tape, and network. The -t type option will display items from type. Types are cpu, fpu, dcache, icache, memory, and qic. The -s option, when used with either the -c or -t options, suppresses output.

The hinv command, when used with the -c or -t options, will exit with a value of 1, if no item of the specified class or type is present in the hardware inventory table. Otherwise, hinv exits with a value of 0.

SEE ALSO
NAME
hyroute – set the HyperNet routing tables

SYNOPSIS
hyroute interface [ -s ] [ -p ] [ -c ] [ -d ] [ file ]

DESCRIPTION
Hyroute manipulates the HYPERchannel(TM) routing information.

With the -s option, it reads file and sets the system’s database according to
the information in the file (see below). The input file defaults to
/usr/etc/hyroute.conf if one is not specified. If the argument ‘-’ is encoun-
tered, hyroute reads from standard input.

The -c option causes hyroute to compare the system’s current information
to that contained in file.

The -p option causes a digested version of file to be printed.

The -d option causes a "dump" of the system’s table (used for debugging
routing code).

FILE FORMAT
The input file is free format. Comment lines start with a ‘*’ in column one.
Statements end with a semicolon.

direct host dest control access ;

Describes a host that can be directly reached from this adapter. Host is a
host name as listed in the hosts(4) file. dest, control, and access are hexa-
decimal numbers. The data will be sent to HYPERchannel address dest
using a control value of control and an access code of access (see the
adapter manuals for details).

The specified remote adapter and the local adapter must both be connected
to one or more common trunks or connected to trunks that are connected
with with link adapters.

gateway host gate1 gate2 gate3 ... ;

Describes a host that must be reached indirectly through any one of the
gateways indicated on the line. The hosts listed are not gateways in the for-
mal sense (they don’t run the Internet gateway protocols), but are hosts on
the HYPERchannel that can "bridge" between subsections of the HYPER-
channel network.

EXAMPLE

* Sample /usr/etc/hyroute.conf.
*
direct sequoia-hy 2200 1100 0;
direct lassen-hy 6104 1100 0;
direct yosemite-hy 2302 1100 0;
direct glacier-hy 4201 1100 0;
gateway redwood-hy lassen-hy yosemite-hy glacier-hy

SEE ALSO
   hy(7), hosts(4)

FILES
   /dev/hy* Character special file to get to the interface (only has an
                 ioctl entry)
                 /usr/etc/hyroute.conf

NOTE
   HYPERchannel is a registered trademark of Network Systems Corp.
NAME

ifconfig – configure network interface parameters

SYNOPSIS

/usr/etc/ifconfig interface address_family [ address [ dest_address ] ] [ parameters ]
/usr/etc/ifconfig interface [ protocol_family ]

DESCRIPTION

Ifconfig is used to assign an address to a network interface and/or configure network interface parameters. Ifconfig is invoked at boot time from /etc/init.d/network to define the network address of each interface present on a machine; you may also use it once the system is up to redefine an interface’s address or other operating parameters. The interface parameter is a string of the form “name unit”, e.g., “enp0”. (The –i option to netstat(1) displays the interfaces on the machine.)

Since an interface may receive transmissions in differing protocols, each of which may require separate naming schemes, it is necessary to specify the address_family, which may change the interpretation of the remaining parameters. Currently, just the “inet” address family is supported.

For the Internet family, the address is either an Internet address expressed in the Internet standard “dot notation” (see inet(3N)), or a host name present in the hosts(4) file, etc/hosts. (Other hosts databases, such as named and YP, are ignored.)

Only the super-user may modify the configuration of a network interface.

The following parameters may be set with ifconfig:

up

Mark an interface “up”. This may be used to enable an interface after an “ifconfig down.” It happens automatically when setting the first address on an interface. If the interface was reset when previously marked down, the hardware will be re-initialized.

down

Mark an interface “down”. When an interface is marked “down”, the system will not attempt to transmit messages through that interface. If possible, the interface will be reset to disable reception as well. This action does not automatically disable routes using the interface.

arp

Enable the use of the Address Resolution Protocol in mapping between network level addresses and link level addresses (default). This is currently implemented for mapping between Internet addresses and 10Mb/s Ethernet addresses.
-arp

Disable the use of the Address Resolution Protocol.

metric n

Set the routing metric of the interface to n, default 0. The routing metric is used by the routing protocol (routed(1m)). Higher metrics have the effect of making a route less favorable; metrics are counted as addition hops to the destination network or host.

netmask mask

Specify how much of the address to reserve for subdividing networks into sub-networks. The mask includes the network part of the local address and the subnet part, which is taken from the host field of the address. The mask can be specified as a single hexadecimal number with a leading 0x, with a dot-notation Internet address, or with a pseudo-network name listed in the network table networks(4). The mask contains 1’s for the bit positions in the 32-bit address which are to be used for the network and subnet parts, and 0’s for the host part. The mask should contain at least the standard network portion, and the subnet field should be contiguous with the network portion.

broadcast

Specify the address to use to represent broadcasts to the network. The default broadcast address is the address with a host part of all 1’s.

trailers

Request the use of a “trailer” link level encapsulation when sending. If a network interface supports trailers, the system will, when possible, encapsulate outgoing messages in a manner which minimizes the number of memory to memory copy operations performed by the receiver. On networks that support the Address Resolution Protocol (see arp(7P); currently, only 10 Mb/s Ethernet), this flag indicates that the system should request that other systems use trailers when sending to this host. Similarly, trailer encapsulations will be sent to other hosts that have made such requests.

-trailers

Disable the use of a “trailer” link level encapsulation (default).

dstaddr

Specify the address of the correspondent on the other end of a point-to-point link.

debug

Enable driver-dependent debugging code; usually, this turns on extra console error logging.
-debug Disable driver-dependent debugging code.

`ifconfig` displays the current configuration for a network interface when no optional parameters are supplied. If a protocol family is specified, `ifconfig` will report only the details specific to that protocol family.

**DIAGNOSTICS**

Messages indicating the specified interface does not exist, the requested address is unknown, or the user is not privileged and tried to alter an interface’s configuration.

**FILES**

/etc/hosts host-address database

**SEE ALSO**

`netstat(1), network(1M)`
NAME

inetd – Internet "super-server"

SYNOPSIS

/usr/etc/inetd [ configuration file ]

DESCRIPTION

When inetd is started at boot time by /etc/init.d/network, it reads its configuration information from the /usr/etc/inetd.conf and listens for connections on certain internet sockets. When a connection is found on one of its sockets, it decides what service the socket corresponds to, and invokes a program to service the request. After the program is finished, it continues to listen on the socket (except in some cases which will be described below). Essentially, inetd allows running one daemon to invoke several others, reducing load on the system. To specify a different configuration file, put its pathname in the file /etc/config/inetd.options.

Each line of the configuration file must be a valid service or a comment. Comments are denoted by a "#" at the beginning of a line. Each field in a line must be specified, with values for each field separated by a tab or a space. The fields are as follows:

- service name
- socket type
- protocol
- wait/nowait
- user
- server program
- server program arguments

There are three types of services that inetd can start: standard, RPC and TCPMUX. A standard service has a well-known port assigned to it and is listed in /etc/services or the Yellow Pages services map (see services(4)); it may be a service that implements an official Internet standard or is a BSD Unix-specific service. RPC services use the Sun RPC calls as the transport; such services are listed in /etc/rpc or the Yellow Pages rpc map (see rpc(4)). TCPMUX services are nonstandard services that do not have a well-known port assigned to them. They are invoked from inetd when a program connects to the "tcpmux" well-known port and specifies the service name. This feature is useful for adding locally-developed servers.

For standard Internet services, the service name field is the name of a valid service in the services(4) database. For "internal" services (discussed below), the service name must be the official name of the service (that is, the first field in /etc/services). For RPC services, the value of service name field consists of the RPC service name, followed by a slash and either a version number or a range of version numbers (e.g., mountd/1). For TCPMUX services, the value of service name field consists of the string "tcpmux"
followed by a slash and the locally-chosen service name. If the service name begins with a "+", it indicates `inetd` will handle the initial handshake with the requesting program otherwise the service is responsible for the handshake. (The handshake is described in the `inetd` section in the Network Programming chapter in the Network Communications Guide.) The service names "help" and the ones listed in `/etc/services` are reserved. The "help" name causes `inetd` to list available TCMUX services. Private protocols should use a service name that has a high chance of being unique. A good practice is to prefix the service name with the name of your organization. Multiple versions of a protocol can suffix the service name with a protocol version number.

The `socket type` should be one of "stream", "dgram", or "raw", depending on whether the socket is a stream, datagram, or raw socket. TCMUX services must use "stream."

The `protocol` must be a valid protocol as given in `/etc/protocols` (see `protocols(4)`). Examples might be "tcp" or "udp". For RPC services, the field consists of the string "rpc" followed by a slash and the name of the protocol (e.g., rpc/tcp or rpc/udp for an RPC service using the TCP or UDP protocol as a transport mechanism). TCMUX services must use "tcp."

The `wait/nowait` field is applicable to datagram sockets only (other sockets should have a "nowait" field in this space). If a datagram server connects to its peer, freeing the socket so `inetd` can received further messages on the socket, it is said to be a "multi-threaded" server, and should specify "nowait". For datagram servers which process all incoming datagrams on a socket and eventually time out, the server is said to be "single-threaded" and should use a "wait" field. Datagram servers that establishes pseudo-connections must be listed as "wait" in order to avoid a race; e.g., the `tftp` server reads the first packet, creates a new socket, and then forks and exits to allow `inetd` to check for new service requests to spawn new servers. TCMUX services must use "nowait."

The `user` field should contain the user name of the user as whom the server should run. This allows for servers to be given less permission than root.

The `server program` field should contain the pathname of the program which is to be executed by `inetd` when a request is found on its socket. If `inetd` provides this service internally, this field should be "internal". For non-internal services, the arguments to the server program should be just as they normally are, starting with `argv[0]`, which is the name of the program. There is a limit of 11 arguments per program (including `argv[0]`).
Inetd provides several trivial services internally by use of routines within itself. These services are "echo", "discard", "chargen" (character generator), "daytime" (human readable time), "time" (machine readable time, in the form of the number of seconds since midnight, January 1, 1900), and "tcpmux" (service multiplexor). All of these services are TCP-based. For details of these services, consult RFCs 862, 863, 864, 867, 868, and 1078.

Inetd rereads its configuration file when it receives a hangup signal, SIGHUP. Services may be added, deleted or modified when the configuration file is reread. To disable a service, add the comment character "#" at the beginning of the service’s line in inetd.conf and send SIGHUP to inetd with the command

```
/etc/killall -HUP inetd
```

EXAMPLES

Here are several example entries for the various types of services

```
ftp    stream tcp   nowait root /usr/etc/ftpd    ftpd -l
ntalk  dgram udp   wait root /usr/etc/talkd    talkd
rusersd/1 dgram rpc/udp wait root /usr/etc/rpc.rusersd rusersd
tcpmux/+date stream tcp   nowait guest /bin/date date
```

NOTE

If Yellow Pages is enabled on your system, inetd uses the YP equivalents of /etc/services and /etc/rpc. Make sure your YP server’s services and rpc databases contain the entries listed in these files.

SEE ALSO

ftpd(1M), rexecd(1M), rlogind(1M), rshd(1M), telnetd(1M), tftp(1M).

"Network Programming" chapter in the Network Communications Guide.

RFCs are available from the Network Information Center at SRI International, Menlo Park, CA.
NAME

infocmp – compare or print out terminfo descriptions

SYNOPSIS


DESCRIPTION

infocmp can be used to compare a binary terminfo(4) entry with other terminfo entries, rewrite a terminfo(4) description to take advantage of the use= terminfo field, or print out a terminfo(4) description from the binary file (term(4)) in a variety of formats. In all cases, the boolean fields will be printed first, followed by the numeric fields, followed by the string fields.

Default Options

If no options are specified and zero or one termnames are specified, the –I option will be assumed. If more than one termname is specified, the –d option will be assumed.

Comparison Options [-d] [-c] [-n]

infocmp compares the terminfo(4) description of the first terminal termname with each of the descriptions given by the entries for the other terminal’s termnames. If a capability is defined for only one of the terminals, the value returned will depend on the type of the capability: F for boolean variables, –I for integer variables, and NULL for string variables.

–d produce a list of each capability that is different. In this manner, if one has two entries for the same terminal or similar terminals, using infocmp will show what is different between the two entries. This is sometimes necessary when more than one person produces an entry for the same terminal and one wants to see what is different between the two.

–c produce a list of each capability that is common between the two entries. Capabilities that are not set are ignored. This option can be used as a quick check to see if the –u option is worth using.

–n produce a list of each capability that is in neither entry. If no termnames are given, the environment variable TERM will be used for both of the termnames. This can be used as a quick check to see if anything was left out of the description.

Source Listing Options [-I] [-L] [-C] [-r]

The –I, –L, and –C options will produce a source listing for each terminal named.
use the `terminfo(4)` names

- `L` use the long C variable name listed in `<term.h>`
- `C` use the `termcap` names
- `r` when using `-C`, put out all capabilities in `termcap` form

If no `termnames` are given, the environment variable `TERM` will be used for the terminal name.

The source produced by the `-C` option may be used directly as a `termcap` entry, but not all of the parameterized strings may be changed to the `termcap` format. `infocomp` will attempt to convert most of the parameterized information, but that which it doesn’t will be plainly marked in the output and commented out. These should be edited by hand.

All padding information for strings will be collected together and placed at the beginning of the string where `termcap` expects it. Mandatory padding (padding information with a trailing `'/') will become optional.

All `termcap` variables no longer supported by `terminfo(4)`, but which are derivable from other `terminfo(4)` variables, will be output. Not all `terminfo(4)` capabilities will be translated; only those variables which were part of `termcap` will normally be output. Specifying the `-r` option will take off this restriction, allowing all capabilities to be output in `termcap` form.

Note that because padding is collected to the beginning of the capability, not all capabilities are output, mandatory padding is not supported, and `termcap` strings were not as flexible, it is not always possible to convert a `terminfo(4)` string capability into an equivalent `termcap` format. Not all of these strings will be able to be converted. A subsequent conversion of the `termcap` file back into `terminfo(4)` format will not necessarily reproduce the original `terminfo(4)` source.

Some common `terminfo` parameter sequences, their `termcap` equivalents, and some terminal types which commonly have such sequences, are:

<table>
<thead>
<tr>
<th><code>terminfo</code></th>
<th><code>termcap</code></th>
<th>Representative Terminals</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>%p1%c</code></td>
<td><code>%.</code></td>
<td>adm</td>
</tr>
<tr>
<td><code>%p1%d</code></td>
<td><code>%d</code></td>
<td>hp, ANSI standard, vt100</td>
</tr>
<tr>
<td><code>%p1%’x’%+%c</code></td>
<td><code>%+x</code></td>
<td>concept</td>
</tr>
<tr>
<td><code>%i</code></td>
<td><code>%i</code></td>
<td>ANSI standard, vt100</td>
</tr>
<tr>
<td><code>%p1%?’x’%&gt;%t%p1’y’%+%;</code></td>
<td><code>&gt;%xy</code></td>
<td>concept</td>
</tr>
<tr>
<td><code>%p2</code> is printed before <code>%p1</code></td>
<td><code>%r</code></td>
<td>hp</td>
</tr>
</tbody>
</table>

April 1990 - 2 - Version 5.0
Use= Option [-u]

- u produce a `terminfo(4)` source description of the first terminal `termname` which is relative to the sum of the descriptions given by the entries for the other terminals `termnames`. It does this by analyzing the differences between the first `termname` and the other `termnames` and producing a description with `use=` fields for the other terminals. In this manner, it is possible to retrofit generic `terminfo` entries into a terminal's description. Or, if two similar terminals exist, but were coded at different times or by different people so that each description is a full description, using `infocmp` will show what can be done to change one description to be relative to the other.

A capability will get printed with an at-sign (@) if it no longer exists in the first `termname`, but one of the other `termname` entries contains a value for it. A capability's value gets printed if the value in the first `termname` is not found in any of the other `termname` entries, or if the first of the other `termname` entries that has this capability gives a different value for the capability than that in the first `termname`.

The order of the other `termname` entries is significant. Since the `terminfo` compiler `tic(1M)` does a left-to-right scan of the capabilities, specifying two `use=` entries that contain differing entries for the same capabilities will produce different results depending on the order that the entries are given in. `infocmp` will flag any such inconsistencies between the other `termname` entries as they are found.

Alternatively, specifying a capability after a `use=` entry that contains that capability will cause the second specification to be ignored. Using `infocmp` to recreate a description can be a useful check to make sure that everything was specified correctly in the original source description.

Another error that does not cause incorrect compiled files, but will slow down the compilation time, is specifying extra `use=` fields that are superfluous. `infocmp` will flag any other `termname` `use=` fields that were not needed.

Other Options [-s dlllllc] [-v] [-V] [-l] [-w width]

- s sort the fields within each type according to the argument below:
  - d leave fields in the order that they are stored in the `terminfo` database.
  - i sort by `terminfo` name.
sort by the long C variable name.
csort by the termcap name.

If no -s option is given, the fields printed out will be sorted alphabetically by the terminfo name within each type, except in the case of the -C or the -L options, which cause the sorting to be done by the termcap name or the long C variable name, respectively.

-v print out tracing information on standard error as the program runs.

-V print out the version of the program in use on standard error and exit.

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

-w change the output to width characters.

Changing Databases [-A directory] [-B directory]
The location of the compiled terminfo(4) database is taken from the environment variable TERMINFO. If the variable is not defined, or the terminal is not found in that location, the system terminfo(4) database, usually in /usr/lib/terminfo, will be used. The options -A and -B may be used to override this location. The -A option will set TERMINFO for the first termname and the -B option will set TERMINFO for the other termnames. With this, it is possible to compare descriptions for a terminal with the same name located in two different databases. This is useful for comparing descriptions for the same terminal created by different people. Otherwise the terminals would have to be named differently in the terminfo(4) database for a comparison to be made.

FILES
/usr/lib/terminfo/*/ compiled terminal description database

DIAGNOSTICS
malloc is out of space!

There was not enough memory available to process all the terminal descriptions requested. Run infocmp several times, each time including a subset of the desired termnames.

use= order dependency found:

A value specified in one relative terminal specification was different from that in another relative terminal specification.
'use=term' did not add anything to the description.
A relative terminal name did not contribute anything to
the final description.

must have at least two terminal names for a comparison to be done.
The –u, –d and –c options require at least two terminal
names.

SEE ALSO
ttic(1M), curses(3X), term(4), terminfo(4) in the Programmer's Reference
Manual.
Chapter 10 of the Programmer's Guide.

NOTE
The termcap database (from earlier releases of UNIX System V) may not be
supplied in future releases.
NAME
init, telinit – process control initialization

SYNOPSIS
/etc/init [ 0123456sSq ]
/etc/telinit [ 0123456sSQqabc ]

DESCRIPTION
Init

init is a general process spawner. Its primary role is to create processes from information stored in the file /etc/inittab (see inittab(4)). This file usually has init spawn getty’s on each line that a user may log in on. It also controls autonomous processes required by any particular system.

init considers the system to be in a run-level at any given time. A run-level can be viewed as a software configuration of the system where each configuration allows only a selected group of processes to exist. The processes spawned by init for each of these run-levels is defined in the inittab file. init can be in one of eight run-levels, 0–6 and S or s. The run-level is changed by having a privileged user run /etc/init. This user-spawned init sends appropriate signals to the original init spawned by the operating system when the system was rebooted, telling it which run-level to change to.

init is invoked inside the UNIX system as the last step in the boot procedure. First init looks in /etc/inittab for the initdefault entry (see inittab(4)). If there is one, init uses the run-level specified in that entry as the initial run-level to enter. If this entry is not in /etc/inittab, init requests that the user enter a run-level from the virtual system console, /dev/console. If an S or an s is entered, init goes into the SINGLE USER state. This is the only run-level that doesn’t require the existence of a properly formatted /etc/inittab file. If it doesn’t exist, then by default the only legal run-level that init can enter is the SINGLE USER state. In the SINGLE USER state the virtual console terminal /dev/console is opened for reading and writing and the command /bin/su is invoked immediately. To exit from the SINGLE USER state, use either init or telinit, to signal init to change the run-level of the system. Note that if the shell is terminated (via an end-of-file), init will only re-initialize to the SINGLE USER state.

When attempting to boot the system, failure of init to prompt for a new run-level may be due to the fact that the device /dev/console is linked to a device other than the physical system console (/dev/tty). If this occurs, init can be forced to relink /dev/console by typing a delete on the system console which is colocated with the processor.
When *init* prompts for the new run-level, the operator may enter only one of the digits 0 through 6 or the letters S or s. If S or s is entered, *init* operates as previously described in the SINGLE USER state with the additional result that /dev/console is linked to the user’s terminal line, thus making it the virtual system console. A message is generated on the physical console, /dev/contty, saying where the virtual terminal has been relocated.

When *init* comes up initially and whenever it switches out of SINGLE USER state to normal run states, it sets the ioctl(2) states of the virtual console, /dev/console, to those modes saved in the file /etc/ioctl.syscon. This file is written by *init* whenever the SINGLE USER state is entered.

If a 0 through 6 is entered *init* enters the corresponding run-level. Any other input will be rejected and the user will be re-prompted.

If this is the first time *init* has entered a run-level other than SINGLE USER, *init* first scans inittab for special entries of the type boot and bootwait. These entries are performed, providing the run-level entered matches that of the entry before any normal processing of inittab takes place. In this way any special initialization of the operating system, such as mounting file systems, can take place before users are allowed onto the system. The inittab file is scanned to find all entries that are to be processed for that run-level.

Run-level 2 is defined to contain all of the terminal processes and daemons that are spawned in the multi-user environment. Hence, it is commonly referred to as the MULTI-USER state. Run-levels 3 and 4 are available to be defined as alternative multi-user environment configurations; however, they are not necessary for system operation and are usually unused.

In a MULTI-USER environment, the inittab file is set up so that *init* will create a process for each terminal on the system that the administrator sets up to respawn.

For terminal processes, ultimately the shell will terminate because of an end-of-file either typed explicitly or generated as the result of hanging up. When *init* receives a signal telling it that a process it spawned has died, it records the fact and the reason it died in /etc/utmp and /etc/wtmp if it exists (see who(1)). A history of the processes spawned is kept in /etc/wtmp.

To spawn each process in the inittab file, *init* reads each entry and for each entry that should be respawned, it forks a child process. After it has spawned all of the processes specified by the inittab file, *init* waits for one of its descendant processes to die, a powerfail signal, or until *init* is signaled by *init* or telinit to change the system’s run-level. When one of these conditions occurs, *init* re-examines the inittab file. New entries can be added to the inittab file at any time; however, *init* still waits for one of the above three conditions to occur. To get around this, *init* Q or *init* q command
wakes init to re-examine the initab file immediately.

If init receives a powerfail signal (SIGPWR) it scans initab for special entries of the type powerfail and powerwait. These entries are invoked (if the run-levels permit) before any further processing takes place. In this way init can perform various cleanup and recording functions during the powerdown of the operating system. Note that in the SINGLE-USER state only powerfail and powerwait entries are executed.

When init is requested to change run-levels (via telinit), init sends the warning signal (SIGTERM) to all processes that are undefined in the target run-level. init waits 5 seconds before forcibly terminating these processes via the kill signal (SIGKILL).

Telinit

Telinit, which is linked to /etc/init, is used to direct the actions of init. It takes a one-character argument and signals init via the kill system call to perform the appropriate action. The following arguments serve as directives to init.

0–6        tells init to place the system in one of the run-levels 0–6.

a,b,c      tells init to process only those /etc/initab file entries having the a, b or c run-level set. These are pseudostates, which may be defined to run certain commands, but which do not cause the current run-level to change.

Q,q        tells init to re-examine the /etc/initab file.

s,S        tells init to enter the single user environment. When this level change is effected, the virtual system teletype, /dev/console, is changed to the terminal from which the command was executed.

FILES

/etc/initab
/etc/utmp
/etc/wtmp
/etc/ioscp.syscon
/dev/console
/dev/contty

SEE ALSO

getty(1M), termio(7).
DIAGNOSTICS
If init finds that it is respawning an entry from /etc/inittab more than 10 times in 2 minutes, it will assume that there is an error in the command string in the entry, and generate an error message on the system console. It will then refuse to respawn this entry until either 5 minutes has elapsed or it receives a signal from a user-spawned init (telinit). This prevents init from eating up system resources when someone makes a typographical error in the inittab file or a program is removed that is referenced in the inittab.

WARNINGS
Telinit and init can be run only by someone who is super-user.

The S or s state must not be used indiscriminately in the /etc/inittab file. A good rule to follow when modifying this file is to avoid adding this state to any line other than the initdefault.

The change to /etc/gettydefs described in the WARNINGS section of the gettydefs(4) manual page will permit terminals to pass 8 bits to the system as long as the system is in multi-user state (run level greater than 1). When the system changes to single-user state, the getty is killed and the terminal attributes are lost. To permit a terminal to pass 8 bits to the system in single-user state, after you are in single-user state, type:

    stty -istrip cs8

BUGS
Attempting to relink /dev/console with /dev/contty by typing a delete on the system console does not work.
NAME

inst – software installation tool

SYNOPSIS

inst [ -f source ] [ -r root ] [ -m machvalue ] ...

DESCRIPTION

inst is the installation tool for distributed software. By default, it will read
from /dev/nrtape, but can read from other local or remote files or tape dev-
ices. It is invoked automatically during a miniroot installation, and can be
explicitly invoked under IRIX for some software. The miniroot is a small
IRIX system that is copied from the distribution onto the disk, and then
booted in such a way that the normal / and /usr file systems are not active.
The normal file systems are then mounted by inst, usually as /root and
/root/usr, so that they can be accessed during installation.

Inst can be invoked as a normal IRIX command, but only certain software
products can be installed on the system while it is running. Root permis-
sions are required to invoke inst in this manner. Not all software can be
installed on the running system; in some cases it is unsafe or technically
infeasible to do so, and inst will not permit such subsystems to be installed.
(For example, the IRIX system itself cannot be updated while it is running.)

Options:

-r root
Set the effective root directory under which the new
software will be installed. By default, this is /root during
miniroot installations, and / under IRIX. Other directories
can be treated as the root of a virtual IRIX tree for diskless
prototype trees, and as a method of installing software some-
where other than over the running IRIX system.

-f source
The default distribution source is /dev/nrtape, but can be
over-ridden with this option, or with explicit "from" com-
mands while running inst.

-m machval
Some software products contain multiple versions of the
same file. Only one of them will be installed, based on the
machine-specific values. By default, the values of CPU-
BOARD and GFXBOARD are set automatically, based on
the physical configuration of the machine. It may in some
cases be necessary to explicitly set one or more machine-
specific values to override the defaults.

Inst has an extensive help command that can serve to answer specific ques-
tions about the installation process. A complete description of the pro-
cedures involved is found in the release notes for the software you are ins-
talling.
FILES
/usr/lib/inst/*  Installation history and supporting files

SEE ALSO
  versions(1m).
NAME
killall – kill named processes

SYNOPSIS
/etc/killall [ [-] signal ]
/etc/killall [ -g ] [ -v ] [ [-]signal ] [ pname ...]
/etc/killall [ -g ] [ -v ] [ -signame ] [ pname ...]
/etc/killall -l

DESCRIPTION
killall is used by /etc/shutdown to kill all active processes not directly related to the shutdown procedure, i.e., not in the same process group.

killall with no options terminates all processes so that the mounted file systems will be unbusied and can be unmounted.

killall sends signal (see kill(1)) to all processes not belonging to the above group of exclusions. The signal number can be preceded by a hyphen ("-"), which is required if pname looks like a signal number. Like kill(1), a mnemonic name for the signal can be used; killall -l lists the signal names. For example:

    killall 16 myproc
    killall -16 myproc
    killall -USR1 myproc

are equivalent. If no signal value is specified, a default of 9 (KILL) is used.

If pname(s) are specified, then only processes with the given names are sent signals. The -v option reports if the signal was sucessfully sent. The -g option causes the signal to be sent to the named processes' entire process group. In this form, the signal number should be preceded by "-" in order to disambiguate it from a process name.

This command can be quite useful for killing a process without knowing its process ID. Killall can be used to stop a run-away user program, without having to wait for ps(1) to find its process ID. It can be particularly useful in scripts, since it makes unnecessary running the output of ps(1) thru grep(1) and then thru sed(1) or awk(1).

FILES
/etc/shutdown

SEE ALSO
fuser(1M), shutdown(1M).
WARNINGS

The first form of *killall* can be run only by the super-user.
NAME
  labelit – provide labels for file systems

SYNOPSIS
  /etc/labelit special [ fsname volume [ -n ] ]

DESCRIPTION
  labelit can be used to provide labels for unmounted disk file systems or file
  systems being copied to tape. You must be superuser to run labelit. The -n
  option provides for initial labeling only (this destroys previous contents).
  With the optional arguments omitted, labelit prints current label values.
  The special name should be the physical disk section (e.g.,
  /dev/dsk/dks0d1s0), or the cartridge tape (e.g., /dev/tape). The device may
  not be on a remote machine.
  The fsname argument represents the mounted name (e.g., root, u1, etc.) of
  the file system.
  Volume may be used to equate an internal name to a volume name applied
  externally to the disk pack, diskette or tape.
  For file systems on disk, fsname and volume are recorded in the superblock.

SEE ALSO
NAME
lboot – configure bootable kernel

SYNOPSIS
/usr/sbin/lboot [-m master] [-s system] [-b boot] [-u unix]

DESCRIPTION
The lboot command is used to configure a bootable UNIX kernel. Master files in the directory master contain configuration information used by lboot when creating a kernel. The file system is used by lboot to determine which modules are to be configured into the kernel.

If a module in master is specified in the system file via "INCLUDE:", that module will be included in the bootable kernel. For all included modules, lboot searches the boot directory for an object file with the same name as the file in master, but with a ".o" or ".a" appended. If found, this object is included when building the bootable kernel.

For every module in the system file specified via "VECTOR:", lboot takes actions to determine if a hardware device corresponding to the specified module exists. Generally, the action is a memory read at a specified base, of the specified size. If the read succeeds, the device is assumed to exist, and its module will also be included in the bootable kernel.

Master files that are specified in the system file via "EXCLUDE:" are also examined; stubs are created for routines specified in the excluded master files that are not found in the included objects.

Master files that are specified in the system file via "USE:" are treated as though the file were specified via the "INCLUDE:" directive, if an object file corresponding to the master file is found in the boot directory. If no such object file is found, "USE:" is treated as "EXCLUDE:".

To create the new bootable object file, the applicable master files are read and the configuration information is extracted and compiled. The output of this compilation is then linked with all included object files.

The options are:

-m master This option specifies the directory containing the master files to be used for the bootable kernel. The default master directory is $ROOT/usr/sysgen/master.d.

-s system This option specifies the name of the system file. The default system file is $ROOT/usr/sysgen/system.

-b boot This option specifies the directory where object files are to be found. The default boot directory is $ROOT/usr/sysgen/boot.
-r ROOT

If this option is specified, ROOT becomes the starting pathname when finding files of interest to lboot. Note that this option sets ROOT as the search path for include files used to generate the target kernel. If this option is not specified, the ROOT environment variable (if any) is used instead.

-p toolpath

This option specifies that the tools (compiler, linker, etc.) used by lboot to generate a new kernel are to be found in the directory toolpath.

-v

This option makes lboot slightly more verbose.

-u unix

This option specifies the name of the target kernel. By default, it is unix.new, unless the -t option is used, in which case the default is unix.install.

-d

This option displays debugging information about the devices and modules put in the kernel.

-t

This option tests if the existing kernel is up-to-date. If the kernel is not up-to-date, it prompts you to proceed. It compares the modification dates of the system file, the object files in the boot directory, and the configuration files in the master directory with that of the existing kernel. It also "probes" for the devices specified with "VECTOR:" lines in the system file. If the devices have been added or removed, or if the kernel is out-of-date, it builds a new kernel, adding ".install" to the target name.

EXAMPLE

lboot -s newsystem

This will read the file named newsystem to determine which objects should be configured into the bootable object.

FILES

/usr/sysgen/system
/usr/sysgen/master.d/*
/usr/sysgen/boot/*

SEE ALSO

NAME

link, unlink – link and unlink files and directories

SYNOPSIS

/etc/link file1 file2
/etc/unlink file

DESCRIPTION

The link command is used to create a file name that points to another file. Linked files and directories can be removed by the unlink command; however, it is strongly recommended that the rm(1) and rmdir(1) commands be used instead of the unlink command.

The only difference between ln(1) and link(1M) is that the latter simply makes the link(2) system call with the arguments specified. No error checking is performed. Analogously, unlink(1M) simply calls the unlink(2) system call with the specified pathname.

SEE ALSO

ln(1), rm(1) and rmdir(1) in the User's Reference Manual.
link(2), unlink(2) in the Programmer's Reference Manual.

WARNINGS

Using the link(1M) command, the super-user can create multiple hard links to the same directory. This is not possible using the normal ln(1) command. It is also not recommended because it creates a file system that is no longer a tree in the strict sense. In such a file system, pwd(1) can give unexpected results when executed in a subdirectory of a directory with that has multiple hard links (other than the usual '.' and '..' entries).

Use of these commands is discouraged. The commands ln(1), rm(1) and rmdir(1) provide the necessary functionality in a safer manner.
NAME
lpadmin – configure the LP spooling system

SYNOPSIS
/usr/lib/lpadmin -p printer [ options ]
/usr/lib/lpadmin -x dest
/usr/lib/lpadmin -d[dest]

DESCRIPTION
lpadmin configures line printer (LP) spooling systems to describe printers, classes and devices. It is used to add and remove destinations, change membership in classes, change devices for printers, change printer interface programs and to change the system default destination. lpadmin may not be used when the LP scheduler, lpsched(1M), is running, except where noted below.

Exactly one of the -p, -d or -x options must be present for every legal invocation of lpadmin.

-pprinter names a printer to which all of the options below refer. If printer does not exist then it will be created.

-xdest removes destination dest from the LP system. If dest is a printer and is the only member of a class, then the class will be deleted, too. No other options are allowed with -x.

-d[dest] makes dest, an existing destination, the new system default destination. If dest is not supplied, then there is no system default destination. This option may be used when lpsched(1M) is running. No other options are allowed with -d.

The following options are only useful with -p and may appear in any order. For ease of discussion, the printer will be referred to as P below.

-cclass inserts printer P into the specified class. Class will be created if it does not already exist.

eprinter copies an existing printer’s interface program to be the new interface program for P.

-h indicates that the device associated with P is hardwired. This option is assumed when adding a new printer unless the -1 option is supplied.

-interface establishes a new interface program for P. Interface is the path name of the new program.
-l indicates that the device associated with \( P \) is a login terminal. The LP scheduler, \texttt{lpsched}, disables all login terminals automatically each time it is started. Before re-enabling \( P \), its current device should be established using \texttt{lpadmin}.

-\texttt{mmodel} selects a model interface program for \( P \). \textit{Model} is one of the model interface names supplied with the LP Spooling Utilities (see \textit{Models} below).

-\texttt{rclass} removes printer \( P \) from the specified class. If \( P \) is the last member of the class, then the class will be removed.

-\texttt{vdevice} associates a new device with printer \( P \). Device is the path-name of a file that is writable by \texttt{lp}. Note that the same device can be associated with more than one printer. If only the \texttt{-p} and \texttt{-v} options are supplied, then \texttt{lpadmin} may be used while the scheduler is running.

Restrictions.

When creating a new printer, the \texttt{-v} option and one of the \texttt{-e}, \texttt{-i} or \texttt{-m} options must be supplied. Only one of the \texttt{-e}, \texttt{-i} or \texttt{-m} options may be supplied. The \texttt{-h} and \texttt{-l} keyletters are mutually exclusive. Printer and class names may be no longer than 14 characters and must consist entirely of the characters A-Z, a-z, 0-9 and _ (underscore).

Models.

Model printer interface programs are supplied with the LP Spooling Utilities. They are shell procedures which interface between \texttt{lpsched} and devices. All models reside in the directory /usr/spool/lp/model and may be used as is with \texttt{lpadmin} \texttt{-m}. Copies of model interface programs may also be modified and then associated with printers using \texttt{lpadmin} \texttt{-i}. The following describes the models which may be given on the \texttt{lp} command line using the \texttt{-o} keyletter:

LQP-40 Letter quality printer using XON/XOFF protocol at 9600 baud.

DQP-10

Dot matrix draft quality printer using XON/XOFF protocol at 9600 baud.

EXAMPLES

1. For a DQP-10 printer named c18, it will use the DQP-10 model interface after the command:

   \texttt{/usr/lib/lpadmin -pc18 -mdqp10}

2. A LQP-40 printer called pr1 can be added to the \texttt{lp} configuration with the command:

April 1990   - 2 -   Version 5.0
/usr/lib/ladmin -ppr1 -v/dev/contty -mlqp40

FILES
/usr/spool/lp/*

SEE ALSO
accept(1M), lpsched(1M).
NAME
lpc – line printer control program

SYNOPSIS
/etc/lpc [ command [ argument ... ] ]

DESCRIPTION
Lpc is used by the system administrator to control the operation of the line printer system. For each line printer configured in /etc/printcap, lpc may be used to:

- disable or enable a printer,
- disable or enable a printer’s spooling queue,
- rearrange the order of jobs in a spooling queue,
- find the status of printers, and their associated spooling queues and printer daemons.

Without any arguments, lpc will prompt for commands from the standard input. If arguments are supplied, lpc interprets the first argument as a command and the remaining arguments as parameters to the command. The standard input may be redirected causing lpc to read commands from file. Commands may be abbreviated; the following is the list of recognized commands.

? [ command ... ]
help [ command ... ]
Print a short description of each command specified in the argument list, or, if no arguments are given, a list of the recognized commands.

abort [ all | printer ... ]
Terminate an active spooling daemon on the local host immediately and then disable printing (preventing new daemons from being started by lpr) for the specified printers.

clean [ all | printer ... ]
Remove any temporary files, data files, and control files that cannot be printed (i.e., do not form a complete printer job) from the specified printer queue(s) on the local machine.

disable [ all | printer ... ]
Turn the specified printer queues off. This prevents new printer jobs from being entered into the queue by lpr.
down { all | printer } message...
Turn the specified printer queue off, disable printing and put message in the printer status file. The message doesn’t need to be quoted, the remaining arguments are treated like echo(1). This is normally used to take a printer down and let others know why (lpq will indicate the printer is down and print the status message).

enable { all | printer ... }
Enable spooling on the local queue for the listed printers. This will allow lpr to put new jobs in the spool queue.

exit
quit
Exit from lpc.

restart { all | printer ... }
Attempt to start a new printer daemon. This is useful when some abnormal condition causes the daemon to die unexpectedly leaving jobs in the queue. Lpq will report that there is no daemon present when this condition occurs. If the user is the super-user, try to abort the current daemon first (i.e., kill and restart a stuck daemon).

start { all | printer ... }
Enable printing and start a spooling daemon for the listed printers.

status { all | printer ... }
Display the status of daemons and queues on the local machine.

stop { all | printer ... }
Stop a spooling daemon after the current job completes and disable printing.

topq printer [ jobnum ... ] [ user ... ]
Place the jobs in the order listed at the top of the printer queue.

up { all | printer ... }
Enable everything and start a new printer daemon. Undoes the effects of down.

DIAGNOSTICS
?Ambiguous command abbreviation matches more than one command
?Invalid command no match was found
?Privileged command command can be executed by root only

FILES
/etc/printcap printer description file
/usr/spool/* spool directories
/usr/spool/*/lock lock file for queue control

April 1990 - 2 - Version 5.0
SEE ALSO
lpd(1M), lpr(1), lpq(1), lpqm(1), printcap(4)
NAME
lpd – line printer daemon

SYNOPSIS
/usr/lib/lpd [-l] [ port # ]

DESCRIPTION
Lpd is the line printer daemon (spool area handler) and is normally invoked
at boot time from the rc(8) file. It makes a single pass through the
printcap(4) file to find out about the existing printers and prints any files
left after a crash. It then uses the system calls listen(2) and accept(2)
to receive requests to print files in the queue, transfer files to the spooling area,
display the queue, or remove jobs from the queue. In each case, it forks a
child to handle the request so the parent can continue to listen for more
requests. The Internet port number used to rendezvous with other processes
is normally obtained with getservbyname(3) but can be changed with the
port# argument. The -l flag causes lpd to log valid requests received from
the network. This can be useful for debugging purposes.

Access control is provided by two means. First, All requests must come
from one of the machines listed in the file /etc/hosts.equiv or /etc/hosts.lpd.
Second, if the "rs" capability is specified in the printcap entry for the
printer being accessed, lpr requests will only be honored for those users
with accounts on the machine with the printer.

The file minfree in each spool directory contains the number of disk blocks
to leave free so that the line printer queue won’t completely fill the disk.
The minfree file can be edited with your favorite text editor.

The file lock in each spool directory is used to prevent multiple daemons
from becoming active simultaneously, and to store information about the
daemon process for lpr(1), lpq(1), and lprm(1). After the daemon has suc-
cessfully set the lock, it scans the directory for files beginning with cf.
Lines in each cf file specify files to be printed or non-printing actions to be
performed. Each such line begins with a key character to specify what to
do with the remainder of the line.

J Job Name. String to be used for the job name on the burst page.

C Classification. String to be used for the classification line on the
burst page.

L Literal. The line contains identification info from the password file
and causes the banner page to be printed.

T Title. String to be used as the title for pr(1).
Host Name. Name of the machine where lpr was invoked.

Person. Login name of the person who invoked lpr. This is used to verify ownership by lprm.

Send mail to the specified user when the current print job completes.

Formatted File. Name of a file to print which is already formatted.

Like "f" but passes control characters and does not make page breaks.

Name of a file to print using pr(1) as a filter.

Troff File. The file contains troff(1) output (cat phototypesetter commands).

Ditroff File. The file contains device independent troff output.

DVI File. The file contains Tex(1) output (DVI format from Stanford).

Graph File. The file contains data produced by plot(3X).

Cfplot File. The file contains data produced by cfplot.

The file contains a raster image.

The file contains text data with FORTRAN carriage control characters.

Name of the font file to use instead of the default.

Name of the font file to use instead of the default.

Name of the font file to use instead of the default.

Name of the font file to use instead of the default.

Width. Changes the page width (in characters) used by pr(1) and the text filters.

Indent. The number of characters to indent the output by (in ascii).

Unlink. Name of file to remove upon completion of printing.

File name. The name of the file which is being printed, or a blank for the standard input (when lpr is invoked in a pipeline).

If a file cannot be opened, a message will be logged via syslog(3) using the LOG_LPR facility. Lpd will try up to 20 times to reopen a file it expects to be there, after which it will skip the file to be printed.
Lpd uses flock(2) to provide exclusive access to the lock file and to prevent multiple daemons from becoming active simultaneously. If the daemon should be killed or die unexpectedly, the lock file need not be removed. The lock file is kept in a readable ASCII form and contains two lines. The first is the process id of the daemon and the second is the control file name of the current job being printed. The second line is updated to reflect the current status of lpd for the programs lpq(1) and lprm(1).

FILES
/etc/printcap printer description file
/usr/spool/* spool directories
/usr/spool/*/minfreeminimum free space to leave
/dev/lp* line printer devices
/etc/hosts.equiv lists machine names allowed printer access
/etc/hosts.lpd lists machine names allowed printer access, but not under same administrative control.

SEE ALSO
lp(1), pac(1M), lpr(1), lpq(1), lprm(1), syslog(3), printcap(4)
BSD Line Printer Spooler Manual
NAME
lpsched, lpshut, lpmove – start/stop the LP scheduler and move requests

SYNOPSIS
/usr/lib/lpsched
/usr/lib/lpshut
/usr/lib/lpmove requests dest
/usr/lib/lpmove dest1 dest2

DESCRIPTION
lpsched schedules requests taken by lp(1) for printing on line printers (LP's).

lpshut shuts down the line printer scheduler. All printers that are printing at the time lpshut is invoked will stop printing. Requests that were printing at the time a printer was shut down will be reprinted in their entirety after lpsched is started again.

lpmove moves requests that were queued by lp(1) between LP destinations. This command may be used only when lpsched is not running.

The first form of the command moves the named requests to the LP destination, dest. Requests are request ids as returned by lp(1). The second form moves all requests for destination dest1 to destination dest2. As a side effect, lp(1) will reject requests for dest1.

Note that lpmove never checks the acceptance status (see accept(1M)) for the new destination when moving requests.

FILES
/usr/spool/lp/*

SEE ALSO
accept(1M), lpadmin(1M).
NAME
lvck – check and restore consistency of logical volumes

SYNOPSIS
/etc/lvck [-l lvtabname] [lvx]
/etc/lvck -d
/etc/lvck block_special_filename

DESCRIPTION
Lvck checks the consistency of logical volumes by examining the logical volume labels of devices constituting the volumes. Depending on the invocation, the volumes may also be checked against lvtab entries, see lvtab(4). The default system file /etc/lvtab is normally assumed in this case; an alternate lvtab file may be specified with the -l option.

Invoked without parameters, lvck checks every logical volume for which there is an entry in /etc/lvtab.

Invoked with the name of a logical volume device, eg lv0, lvck checks only that entry in /etc/lvtab.

Invoked with the -d flag, lvck ignores /etc/lvtab and searches through all disks connected to the system in order to locate all logical volumes which are present. lvck prints a description of each logical volume found in a form resembling an lvtab entry; this facilitates recreation of an lvtab for the system should this be necessary.

Invoked with the device block special filename of a disk device (eg /dev/dsk/lp0d1s4), lvck prints any logical volume label which exists for that device, again in a form resembling an lvtab entry. Note that this mode of lvck is purely informational; no checks are made of any other devices mentioned in the label.

Lvck has some repair capabilities. If it determines that the only inconsistency in a logical volume is that a minority of devices have missing or corrupt labels, it is able to restore a consistent logical volume by rewriting good labels. lvck interactively queries the user before attempting any repairs on a volume.

. DIAGNOSTICS
Lvck detects four general types of errors:
1) Disks connected in the wrong place.
2) Inconsistencies between the on-disk labels of a volume.
3) Internal inconsistencies in an lvtab entry.
4) Inconsistencies between a volume defined by its on-disk labels, and the lvtab entry for that volume.

(The two latter are relevant only for modes of lvck which examine the lvtab). Details of these errors are given below.

1) If a disk device which is a member of a logical volume is connected with the wrong id, the volume cannot be used since the device will not be correctly located from the labels. Lvck will print a message describing the problem, for example:

"lvck: device currently connected as /dev/dsk/ips1d2s7
was initialized when connected as /dev/dsk/ips0s2s7.

lvck: Incorrect device connections must be rectified
before logical volumes can be used."

The offending disks must be physically reconnected with the appropriate id. See intro(7) for details of the SGI disk device id conventions.

2) If lvck detects inconsistencies between the on-disk labels for a logical volume, it prints a description of the volume in a form resembling an lvtab entry, with the device pathname for each member on a separate line. A short message describing the problem with the member appears on the line with the pathname. For example:

lv6:test 6:stripes=3:step=31:devs=\n /dev/dsk/ips0d1s2, \n /dev/dsk/ips0d1s3, <CAN'T ACCESS> \n /dev/dsk/ips0d1s4, \n /dev/dsk/ips0d1s11, <NO LABEL PRESENT> \n /dev/dsk/ips0d1s12, \n /dev/dsk/ips0d1s13

Possible messages and their meanings are:

<NOT A MEMBER OF THIS VOLUME>

The label for this device identifies it as a member of a different volume from the other devices. Probably the wrong disk has been connected.
<CAN'T ACCESS>
The named special file is missing or cannot be opened. The
disk may be missing, or there may be a hardware problem.

<CAN'T READ DISK HEADER>
Lvck could not read the disk header partition for this device
in order to search for the logical volume label.

<ILLEGAL PARTITION TYPE>
The pathname refers to a type of partition (such as track
replacement) which is not legal as part of a logical volume.
Probably logical volume labels or disk headers have been
corrupted.

<INCORRECT PARTITION SIZE>
The partition size does not agree with the logical volume
label. Possibly the disk has been incorrectly repartitioned
since creation of the logical volume.

<SPECIAL FILE DEV IS WRONG>
The major and minor numbers of the special file disagree
with the SGI naming conventions. There has probably been
an incorrect use of mknod(1M) in the /dev/dsk directory.

<FILE NOT BLOCK SPECIAL>
The pathname does not refer to a block special file, even
though it has the conventional format. The /dev/dsk directory
needs repair.

<INCORRECT PARTITION TYPE>
The partition type stored in the disk header does not indicate
that this partition is a logical volume member. Probably the
wrong disk has been connected.

<LABEL UNREADABLE>
The disk header directory indicates that a logical volume
label exists for this device, but it cannot be read. Possibly
there is a bad block on the disk.

<NO LABEL PRESENT>
There is no logical volume label for this device. It may have
been inadvertently deleted, or the wrong disk may have been
connected.

<LABEL CORRUPTED>
Label is present but damaged.
3) Internal inconsistencies in lvtab entries.
   In this case lvck prints error messages which are intended to be
   self-explanatory. Possible messages are listed below (the items in
   brackets <xxx> represent places where the actual erring values will
   appear):

   "Lvtab entry with no device name: ignored"

   "Lvtab entry with illegal device name <xxx>: ignored"

   "Illegal number of pathnames <n> in lvtab entry <lvx>: ignored"

   "Number of pathnames in lvtab entry <lvx> is not multiple of strip-
   ing: entry ignored."

   "Illegal striping step in lvtab entry <lvx>: ignored"

   "Duplicate lvtab entry <lvx>: ignored"

   "Bad pathname <xxx> in lvtab entry <lvx>: entry ignored."

   "Duplicate pathname <xxx> in lvtab entry <lvx>: entry ignored"

   "Illegal entry <lvx> in logical volume table: ignored"

   See lvtab(4) for details of the expected form of lvtab entries, and
   constraints upon the entries. Limits on the number of devices in a
   volume, striping step size, volume name length, etc are given in the
   include file <sys/lvtab.h>.

4) Inconsistencies between on-disk labels and the lvtab entry.
   This may occur if the lvtab entry has been incorrectly modified, or if
   the disks connected to the system do not contain the logical volume
   expected in the lvtab entry. Lvck prints error messages which are
   intended to be self-explanatory. Possible messages are listed below
   (the items in brackets <xxx> represent places where the actual
   erring values will appear):

   "Volume name <xxx> in lvtab entry disagrees with name <yyy> in
   on-disk labels."

   "Number of devs in lvtab entry <lvx> is greater than number <d> in
   on-disk labels."
"Number of devs in lvtab entry <lvx> is less than number <d> in on-disk labels."

"Stripes specified in lvtab entry for <lvx> don’t agree with on-disk labels."

"Step specified in lvtab entry for <lvx> doesn’t agree with on-disk labels."

SEE ALSO
lvinit(1M), mklv(1M), lvtab(4), intro(7M), lv(7M).

CAVEAT
The repair capabilities of lvck are limited to recreating damaged or missing logical volume labels for disk devices so that the system is again able to use an ensemble of disk devices as a logical volume. However, if data on the devices themselves has been corrupted, or if an incorrect disk device has been connected, the contents of the logical volume will be corrupt. It is strongly advisable to check that no incorrect disks have been connected before proceeding with any repair attempt.
NAME
Lvinit – initialize logical volume devices.

SYNOPSIS
/etc/lvinit [-l lvtabname] [volume_device_names]

DESCRIPTION
Lvinit initializes the logical volume device driver which allows access to
disk storage as logical volumes. See lv(7m).

It is run automatically on system startup, and will not normally need to be
invoked explicitly.

It works from entries in /etc/lvtab, see lvtab(4).

No data access to a logical volume device is possible until it has been ini-
tialized with lvinit, since the initialization process provides the driver with
the information needed to map requests on the logical volume device into
requests on the underlying physical disk devices.

Note: this implies that the root filesystem of a machine must reside on a reg-
ular partition rather than a logical volume, since lvinit must be accessible
before logical volumes can be initialized.

Invoked without arguments, lvinit initializes every logical volume device
for which there is an entry in /etc/lvtab. The –l option allows an alternate
lvtab file to be specified. If volume_device_name arguments are present it
initializes only those volumes in the lvtab identified by the arguments.
These arguments must be of the form lvn where n is a small integer, for
example lv2. The necessary information about the devices (disk partitions)
constituting the logical volume, and the volume geometry, is obtained from
the logical volume labels for the devices specified in the lvtab entry as con-
stituting the volume.

Note that the constituent devices must have been labeled as members of the
volume with mklv before the volume can be initialized.

DIAGNOSTICS
Lvinit does checks of the lvtab entry and the on-disk labels of a volume
before attempting initialization.

See the DIAGNOSTICS section of the lvck man page for possible error
messages.
SEE ALSO
mklv(1M), lvck(1M), lvtab(4), lv(7M).
NAME
MAKEDEV – Create device special files

SYNOPSIS
/dev/MAKEDEV [alldevs] [mindevs] [links] [owners] [generic] [device]

DESCRIPTION
MAKEDEV creates specified device files in the current directory; it is primarily used for constructing the /dev directory. alldev and owners are assumed.

All devices are created relative to the current directory, so this command is normally executed from /dev. In order to create the devices successfully, you must be the superuser.

The following arguments are recognized by MAKEDEV.

tty  Creates tty (controlling terminal interface) files for CPU serial ports. In addition, creates special files for console, syscon, systty, keybd, mouse, dials, and tablet. See duart(7), console(7), keyboard(7), and mouse(7) for details.

cdio  Creates additional tty files enabled by using the Central Data serial board.

pty  Creates special files to support "pseudo terminals". See pty(7M) for details.

ips  Creates special files for ESDI disks connected to an Interphase ESDI disk controller. See ips(7M) for details.

ipi  Creates special files for IPI disks connected to a XYLOGICS IPI disk controller. See ipi(7M) for details.

dks  Creates special files for SCSI disks. See dksc(7M) for details.

xyl  Creates special files for SMD disks connected to a Xylogics SMD disk controller. See xyl(7M) for details.

qictape  Creates special files for 1/4-inch cartridge tape drives connected to an ISI QIC-O2 tape controller. See ts(7M) for details.

tps  Creates special files for SCSI tape drives. See tps(7M) for details.

magentape  Creates special files for 1/2-inch tape drives connected to a Xylogics Model 772 tape controller. See xmt(7M) for details.
ikon  Creates special files for hardcopy devices connected to an
Ikon 10088 (or 10088A) printer controller. This includes
Versatec TTL and Versatec differential compatible devices, as
well as (Tektronix-compatible) Centronics printers. See ik(7)
for details.

gpib  Creates special files for the National Instruments GPIB-1014
controller and associated devices. See gpib(7) for details.

hl   Creates special files for the hardware spinlock driver to use in
process synchronization (IRIS-4D/GTX models only).

gro  Creates special files for graphics output devices on IRIS-4D
models such as the 4D/50, 60, 60G, 70 and 70G.

grin Creates special files for graphics input devices.

gm   Creates the special file for the logical console device on the
IRIS-4D/GT and 4D/GTX models.

t3270 Creates the special files for the IBM 3270 interface controller.

gse  Creates the special files for the IBM 5080 interface controller.

dn_ll Creates the special file for the 4DDN logical link driver.

dn_netman Creates the special file for the 4DDN network management
driver.

audio Creates the special file for the bi-directional audio channel
interface for the IRIS-4D/20 series. See audio(7) for details.

plp  Creates the special file for the parallel printer interface for the
IRIS-4D/20 series. See plp(7) for details.

generic Creates miscellaneous, commonly used devices: tty, the con-
trolling terminal device; mem, kmem, mmem, and null, the
memory devices; prf, the kernel profiling interface; queue, the
graphics event queue interface; and zero, a source of zeroed
unnamed memory. See tty(7), mem(7), prf(7), and zero(7)
for details concerning these respective devices.

links This option does both disk and tape

disk This option creates all the disk device special files for the dks
ips ipi and xyl drives, and then creates links by which one can
conveniently reference them without knowing the
configuration of the particular machine. The links root, rroot,
swap, rswap, usr, rusr, vh and rvh are created to reference the
current root, swap, usr and volume header partitions.
**tape**

This option creates all the tps and xmt tape devices, then makes links to *tape*, *nrtape*, *tapens*, and *nrtapens* for the first tape drive found, if one exists. It first checks for *xmt*, then for SCSI in target ID order.

**mindevs**

This option is shorthand for creating the *generic*, *links*, *pty*, *ttys gro*, and *grin* device files.

**alldevs**

This option creates all of the device special files listed above.

**owners**

This option changes the owner and group of the files in the current directory to the desired default state.

**onlylinks**

This option does only the link portion of *disk* and *tape* above, in case a different disk is used as root, or a different tape drive is used.

**BUGS**

The links made for */dev/ usr* and */dev/rusr* always point to partition 6 of the root drive. While this is the most common convention, it is not invariable.

If a system has been reconfigured with the */usr* filesystem in some place other than this default, by specifying the device in */etc/fstab*, see *fstab*(4), the */dev/ usr* and */dev/rusr* devices will NOT point to the device holding the real */usr* filesystem.

**SEE ALSO**

*mknod*(1M)
NAME
makewhatis – make manual page "whatis" database for use with apropos

SYNOPSIS
/usr/lib/makewhatis [-M manpath] [filename]

DESCRIPTION
makewhatis scans the manual page trees, parses the manual pages, and
strips out the NAME section information to create the "whatis" database
used by apropos(1), man(1), and whatis(1).

By default, makewhatis creates the file /usr/catman/whatis. Another file
may be created as the database by specifying its filename on the command
line.

An alternate manual page tree may be specified by using the -M option to
specify a path or paths to a manual page tree or trees. See the discussion of
the -M option in man(1) for more details.

The format of the whatis file created is based on that used by William Joy’s
UC Berkeley version of apropos.

ENVIRONMENT
The environment variables used by man(1) apply to makewhatis.

SEE ALSO
apropos(1), man(1), whatis(1)

FILES
/usr/catman/whatis

CAVEATS
This program is S...L...O...W. Expect execution times of about 10-15
minutes. The reason is that EVERY manual page on the system is read.

You must run makewhatis as root.
NAME
  mkboottape – make a boot tape

SYNOPSIS
  /etc/mkboottape [-l] [-f output_file_name] file1 file2 ... file20

DESCRIPTION
  mkboottape builds a special directory which contains the list of file names
  and their logical block number relative to the beginning of the output file.
  There may be no more than twenty files and the file names must be less than
  or equal to sixteen characters. The -l option may be used to list the con-
  tents. In this case the file arguments are ignored. The -f option may be
  used to specify an alternate output file or device. The default is /dev/tape.

  Typical use for this program is to create the output file which is dd’ed to
  tape using a blocking factor of 16K. The prom monitor knows about this
  special directory and can boot any one of the files found in the directory.

  Unless you have means to create a stand alone program this utility is use-
  less.

SEE ALSO
  dd(1)

FILES
  /dev/tape - default output device
NAME
mkcentpr – register a color Centronics-interface printer with LP

SYNOPSIS
mkcentpr [-v] [-l] [-d device] type printer

DESCRIPTION
mkcentpr creates a filter program to control a color printer that communicates with the IRIS through a Centronics parallel interface, then registers the printer with LP. printer is the name you give the printer (see lpadmin(1M)). type is the type of printer being used. You must specify the printer type by providing one of the following keys:

mits for Mitsubishi

tek for the Tektronics 4692

vers for the Versatec

seiko for the Seiko CH-5312

seiko03 for the Seiko CH-5303

kodak for the Kodak SV6500

The filter program will normally keep a log of its activities in the file /usr/spool/lp/log. The -l option causes the filter to keep its log in /usr/spool/lp/etc/log/printer–log instead. The -v option causes the filter to keep a more verbose log. The -d option is used to specify the parallel port to which the printer is connected. There are two currently supported parallel ports: /dev/cent refers to the VME parallel board, and /dev/plp refers to the built-in parallel port on the 4D/20. It is not normally necessary to specify a device since mkcentpr will choose one according to the type of machine being used.

After running this program, you might consider making this printer the default printer using lpadmin(1M). This command is only usable by the system administrator.

FILES
/usr/spool/lp/log
/usr/spool/lp/etc/log/printer–log
/usr/spool/lp/interface/printer
/dev/cent
/dev/plp

SEE ALSO
accept(1M), addclient(1M), enable(1), lp(1), lpadmin(1M), lpsched(1M), lpshut(1M), lpstat(1), mknetpr(1M), rmprinter(1M).

April 1990 - 1 -

Version 5.0
NAME
mkfile – create a file

SYNOPSIS
/etc/mkfile [−v] size[k/b/m] filename...

DESCRIPTION
mkfile creates one or more files, and the file is padded with zeroes by
default. The default size is in bytes, but it can be flagged as kilobytes,
blocks, or megabytes, with the k, b, or m suffixes, respectively.

OPTIONS
−v Verbose. Report the names and sizes of created files.
NAME
mkfs – construct a file system

SYNOPSIS
/etc/mkfs [-q] [-i] [-r] special [proto]
/etc/mkfs [-q] [-i] [-r] special blocks inodes heads sectors cgsise cgsalign ialign [proto]

DESCRIPTION
mkfs constructs a file system by writing on the special file using the values found in the remaining arguments of the command line. Normally mkfs prints the parameters of the filesystem to be constructed; the -q flag suppresses this.

If the -i flag is given, mkfs will ask for confirmation after displaying the parameters of the filesystem to be constructed.

The -r flag causes mkfs to write only the superblock, without touching other areas of the filesystem. See the section below on the recovery option.

When the first form of mkfs is used, mkfs obtains information about the device size and geometry by means of appropriate IOCTLs, and assigns values to the filesystem parameters on the basis of this information.

If the second form of mkfs is used, then all the filesystem parameters must be specified from the command line. Each argument other than special and proto is interpreted as a decimal number.

The filesystem parameters are as follows:

blocks is the number of physical (512 byte) disk blocks the file system will occupy.
inodes is the number of inodes the filesystem should have as a minimum.
heads is an unused parameter, retained only for backward compatibility.
sectors is the number of sectors per track of the physical medium.
cgsise is the size of each cylinder group, in disk blocks, approximately.
cgsalign is the boundary, in disk blocks, that a cylinder group should be aligned to.
ialign is the boundary, in disk blocks, that each cylinder group’s inode list should be aligned to.

Once mkfs has the filesystem parameters it needs it then builds a file system containing two directories. The filesystems root directory is created with one entry, the lost+found directory. The lost+found directory is filled with zeros out to approximately 10 disk blocks so as to allow space for fsck(1M)
to reconnect disconnected files. The boot program block (block zero) is left uninitialized.

If the the optional proto argument is given, mkfs uses it as a prototype file and will take its directions from that file. Note that the blocks and inodes specifiers in the proto file are provided for backwards compatibility, but are otherwise unused. The prototype file contains tokens separated by spaces or new-lines. A sample prototype specification follows (line numbers have been added to aid in the explanation):

1. /stand/diskboot
2. 4872 110
3. d—777 3 1
4. usr d—777 3 1
5. sh ——755 3 1 /bin/sh
6. ken d—755 6 1
7. $
8. b0 b—644 3 1 0 0
9. c0 c—644 3 1 0 0
10. fifo p—644 3 1
11. symlink l—644 3 1 /a/symbolic/link
12. $
13. $

Line 1 in the example is the name of a file to be copied onto block zero as the bootstrap program. This is in fact a dummy argument for backward compatibility; boot blocks are not used on SGI machines, and mkfs merely clears block zero.

Line 2 specifies the number of physical (512 byte) blocks the file system is to occupy and the number of i-nodes in the file system. These values are ignored.

Lines 3-11 tell mkfs about files and directories to be included in this file system.

Line 3 specifies the root directory.

lines 4-6 and 8-11 specifies other directories and files.

The $ on line 7 tells mkfs to end the branch of the file system it is on, and continue from the next higher directory. The $ on lines 12 and 13 end the process, since no additional specifications follow.

File specifications give the mode, the user ID, the group ID, and the initial contents of the file. Valid syntax for the contents field depends on the first character of the mode.
The mode for a file is specified by a 6-character string. The first character specifies the type of the file. The character range is `-bcdpl` to specify regular, block special, character special, directory files, named pipes (fifos) and symbolic links, respectively. The second character of the mode is either `u` or `g` to specify set-user-id mode or not. The third is `g` or `r` for the set-group-id mode. The rest of the mode is a 3 digit octal number giving the owner, group, and other read, write, execute permissions (see `chmod(1)`).

Two decimal number tokens come after the mode; they specify the user and group IDs of the owner of the file.

If the file is a regular file, the next token of the specification may be a path name whence the contents and size are copied. If the file is a block or character special file, two decimal numbers follow which give the major and minor device numbers. If the file is a symbolic link, the next token of the specification is used as the contents of the link. If the file is a directory, `mkfs` makes the entries `.` and `..` and then reads a list of names and (recursively) file specifications for the entries in the directory. As noted above, the scan is terminated with the token `$`.

**RECOVERY OPTION**

The `-r` flag causes `mkfs` to write only the superblock, without touching the remainder of the filesystem space. This allows a last-ditch recovery attempt on a filesystem whose superblocks have been destroyed: by running `mkfs` on the device with the `-r` option, a superblock is created from which `fsck(1M)` can obtain the geometry information it needs to analyze the filesystem.

Note that this procedure will only be of use if the regenerated superblock matches the parameters of the original filesystem. If the filesystem was created using the long form invocation, parameters identical to the original invocation must be given with the `-r` option. Note also that filesystem defaults have changed between IRIX 3.2 and 3.3 to allow more efficient use of disk space; the `-r` option will not be useful for filesystems created under IRIX versions earlier than 3.3.

It should be clear that this is a limited recovery facility; it will not help if (for example) the root directory of the filesystem has been destroyed.

**SEE ALSO**

`chmod(1)` in the *User's Reference Manual.*

**BUGS**

With a prototype file, it is not possible to specify hard links.
NAME
mk lv – construct or extend a logical volume

SYNOPSIS
/etc/mk lv [-l lvtabname] [-f] volume_device_name

DESCRIPTION
Mk lv constructs a logical volume by writing logical volume labels for the
devices which are to constitute the volume.

It works from an entry in /etc/lvtab, see lvtab(4), and constructs the logical
volume identified in /etc/lvtab by the volume_device_name argument. This
argument must be of the form lvn where n is a small integer, for example
lv2. Mk lv obtains The necessary information about the devices (disk parti-
tions) constituting the logical volume, and the volume geometry, from the
lvtab entry.

Note that an existing logical volume may be extended by adding further
device pathnames to the end of the existing /etc/lvtab entry, and then rerun-
nig mk lv on that entry.

After writing the labels, mk lv initializes the logical volume device with the
appropriate information. Effectively this invokes the functionality of
lvinit(1M).

The following options are accepted by mk lv.

-f Normally, for safety, mk lv checks whether any of the specified con-
stituent devices are already part of a logical volume, or appear to
contain a filesystem. These checks are skipped if the -f flag is
given; this is useful for recycling disks which contain obsolete logi-
cal volumes or filesystems.

-l Mk lv normally works from the default system file /etc/lvtab. This
option allows an alternate lvtab file to be specified.

DIAGNOSTICS
1) If the volume_device_name argument is not found in the lvtab mk lv
prints the error message:

"mk lv: <arg> not found in lvtab".

2) The lvtab entry is checked before use. Mk lv prints error messages if
problems are found. See the DIAGNOSTICS section of lvck(1M)
for possible error messages concerned with lvtab entries.
3) *MKLV* checks the specified devices for accessibility and legality before proceeding. If errors are detected *MKLV* prints the lvtab entry, with each device pathname on a separate line. A short message describing the problem with the device appears on the line with the pathname. For example:

```
 lv6: test 6: stripes=3: step=31: dev s= \
   / dev/dsk/fps0d1s2,    \
   / dev/dsk/fps0d2s3, <CAN'T ACCESS> \
   / dev/dsk/fps0d3s4, <WRONG PARTITION SIZE> \
   / dev/dsk/fps1d1s11, <CAN'T READ DISK HEADER> \
   / dev/dsk/fps0d1s8, <ILLEGAL PARTITION TYPE> \
   / dev/dsk/fps0d1s13
```

Possible messages and their meanings are listed in the DIAGNOSTICS section of *lvck(1M)*.

4) If the `-f` flag is not given, *MKLV* checks whether there appears to be a filesystem on any of the specified devices. If so, it prints the warning:

"<pathname> appears to contain a filesystem. Proceed? (y/n)"

and waits for user response before proceeding.

5) If the `-f` flag is not given, *MKLV* checks whether any of the specified devices are already part of a logical volume which is inconsistent with the volume specified in the lvtab entry. (Note: it is legal when extending a volume for the on-disk volume to be a consistent subset of the newly specified volume). If an inconsistency exists, *MKLV* prints the error message:

"devices specified for <lvx> already contain a logical volume which is inconsistent with the volume specified."

**NOTE**

The logical volume labels do not occupy space on the constituent partitions themselves, but are files in the Disk Volume Header partitions of the disks containing the partitions, located via the header directory. See *vh(7M)*. They are named for the partitions to which they refer, having names of the form lvlabn where n is the partition number. Thus, if partition 6 on a disk is part of a logical volume, there will be a logical volume label file named 'lvlab6' in the header partition of that disk.
SEE ALSO

ininit(1M), lvck(1M), lvtab(4), lv(7M).
NAME
mknetpr – provide access to a remote printer

SYNOPSIS
mknetpr [-v] [-l] printer host netprinter

DESCRIPTION
mknetpr creates a filter program that accesses a real printer over the network and registers the "printer" with the LP system. printer is the name you give the printer (see lpadmin(1M)). host is the name of an IRIS that has a real printer attached to it. netprinter is the name of the real printer on the remote machine. If the remote printer is a LaserWriter and you want to print troff(1) files, you need to duplicate much of the software that is on the host machine in order to generate a PostScript file that can be transmitted to the remote printer. Many of the filters that prepare data for the LaserWriter default their output to a printer or class of printers called PostScript. Therefore, it is a good idea to use the name PostScript for the printer. You can override such action using the "destination" switch in the various programs.

The filter program will normally keep a log of its activities in the file /usr/spool/lp/log. The -l option causes the filter to keep its log in /usr/spool/lp/etc/log/<printer>-log instead. The -v option causes the filter to keep a more verbose log.

After running this program, you might consider making this printer the default printer using lpadmin(1M). This command is only usable by the system administrator.

This program installs a printer filter program that appears to the LP system as a real printer filter. The data passed to the filter is transmitted to the host machine, then submitted to the remote printer’s queue using lp.

FILES
/usr/spool/lp/log
/usr/spool/lp/etc/log/<printer>-log
/usr/spool/lp/interface/<printer>

SEE ALSO
accept(1M), addclient(1M), enable(1), lp(1), lpadmin(1M), lpsched(1M), lpshut(1M), lpstat(1), rmprinter(1M).
NAME
mknod – build special file or named pipe (FIFO)

SYNOPSIS
/etc/mknod name b | c major minor
/etc/mknod name p

DESCRIPTION
Mknod makes a directory entry and corresponding i-node for a special file
or named pipe. The first argument is the name of the entry to create.

In the first form of the command, the second argument is b if the special file
describes a block device (disks, tape) or c if it is a character device (e.g. a
tty line). The last two arguments are numbers specifying the major device
number and the minor device number (e.g., unit, drive, or line number).
They may be either decimal or octal. The assignment of major device
numbers is specific to each system. The information is contained in the sys-
tem source file /usr/include/sys/major.h.

You must be the super-user to use this form of the command. The normal
convention is to keep all special device files under the /dev directory.

The second form of the mknod command is used to create named pipes
(a.k.a. FIFOs). This form of the command is not restricted to the super-
user.

SEE ALSO
mknod(2)
NAME
mkPS – register a LaserWriter printer with LP

SYNOPSIS
mkPS [-f] [-v] [-l] name tty

DESCRIPTION
mkPS creates a filter program that controls a LaserWriter printer, then
registers the printer with LP. name is the name you give the printer (see
ladmin(1M)). tty is a tty in /dev that the printer is attached to. The filter
program will normally keep a log of its activities in the file
/usr/spool/lp/log. The -l option causes the filter to keep its log in
/usr/spool/lp/transcript/name-log instead. The -v option causes the filter to
keep a more verbose log. The -f option inhibits automatic page reversal and
should be used with printers that support page reversal in the printer such as
the LaserWriter II.

The following options may be given as printer-dependent options using the
-o option of the lps(1) command:

verbose Make a more verbose log entry for a particular print request.
This has the effect of turning on the -v flag described above
for an individual request.

-h Do not print a banner page for this request.

-m Mail any output from the printer back to the user. Some
PostScript files may result in diagnostic output from the
printer. This will normally be placed in the log file. If the -m
option is specified then the output will be sent via mail(1)
back to the originator.

-r Do not perform page reversal for this print request. Page
reversal is normally performed for PostScript files unless the
-f option is used with mkPS. Using the -r option inhibits this
for just one request.

After running this program, you might consider making this printer the
default printer using lpsadmin(1M). This command is only usable by the
system administrator.

FILES
/usr/spool/lp/log
/usr/spool/lp/transcript/name-log
/usr/spool/lp/interface/name
/dev/ttydn
SEE ALSO

accept(1M), addclient(1M), enable(1), lp(1), lpadmin(1M), lpsched(1M),
lpshut(1M), lpstat(1), mknetpr(1M), rmprinter(1M).
NAME

mount, umount – mount and dismount filesystems

SYNOPSIS

/etc/mount [ -p ]
/etc/mount [ -h host ] [ -fnv ]
/etc/mount -a[cfnv] [ -t type ]
/etc/mount [ -cfnv ] [ -t type ] [ -b list ]
/etc/mount [ -cfnrv ] [ -t type ] [ -o options ] fsname dir
/etc/mount [ -cfnrv ] [ -o options ] fsname 1 dir

/etc/umount -a[kv] [ -t type ]
/etc/umount -h host [ -kv ] [ -b list ]
/etc/umount [ -kv ] fsname 1 dir

DESCRIPTION

mount attaches a named filesystem fsname to the filesystem hierarchy at the
pathname location dir. The directory dir must already exist. It becomes
the name of the newly mounted root. The contents of dir are hidden until the
filesystem is unmounted. If fsname is of the form host:path, the filesystem
type is assumed to be nfs.

umount unmounts a currently mounted filesystem, which can be specified
either as a mounted-on directory or a filesystem.

mount and umount maintain a table of mounted filesystems in /etc/mtab,
described in mtab(4). If invoked without an argument, mount displays the
table. If invoked with only one of fsname or dir, mount searches the file
/etc/fstab (see fstab(4)) for an entry whose dir or fsname field matches the
given argument. For example, if this line is in /etc/fstab:

/dev/usr /usr efs rw 0 0

then the commands mount /usr and mount /dev/usr are shorthand for
mount /dev/usr /usr.

MOUNT OPTIONS

-a Attempt to mount all the filesystems described in /etc/fstab. (In
this case, fsname and dir are taken from /etc/fstab.) If a type is
specified, all of the filesystems in /etc/fstab with that type are
mounted. Filesystems are not necessarily mounted in the order
listed in /etc/fstab.

-b list “all-But” – attempt to mount all of the filesystems listed in
/etc/fstab except for those associated with the directories contained
in list. list consists of one or more directory names separated by
commas.
-c In invoke fsstat(1M) on each filesystem being mounted, and if it indicates that the filesystem is dirty, call fsck(1M) to clean the filesystem. fsck is passed the -D and -y options.

-f Fake a new /etc/mtab entry, but do not actually mount any filesystems.

-h host Mount all filesystems listed in /etc/fstab that are remote-mounted from host.

-n Mount the filesystem without making an entry in /etc/mtab.

-o options Specify options, a list of comma-separated words described in fstab(4).

-p Print the list of mounted filesystems in a format suitable for use in /etc/fstab.

-r Mount the specified filesystem read-only. This is a shorthand for:

    mount -o ro fsname dir

Physically write-protected and magnetic tape filesystems must be mounted read-only, or errors occur when access times are updated, whether or not any explicit write is attempted.

-t type The next argument is the filesystem type. The accepted types are dbg, efs and nfs; see fstab(4) and dbg(4) for a description of these filesystem types.

-v Verbose – mount displays a message indicating the filesystem being mounted and any problems encountered.

UMOUNT OPTIONS

-a Attempt to unmount all the filesystems currently mounted (listed in /etc/mtab). In this case, fsname is taken from /etc/mtab.

-b list "all-But" – attempt to unmount all of the filesystems currently mounted except for those associated with the directories contained in list. list consists of one or more directory names separated by commas.

-h host Unmount all filesystems listed in /etc/mtab that are remote-mounted from host.

-k dir Attempt to kill processes which have open files or current directories in dir then unmount the associated filesystem.

-t type Unmounts all filesystems of a given filesystem type. The accepted types are dbg, efs and nfs.
-v Verbose – **umount** displays a message indicating the filesystem being unmounted and any problems encountered.

**EXAMPLES**

- `mount /dev/usr /usr` mount a local disk
- `mount -avt efs` mount all efs filesystems and be verbose
- `mount -t nfs server:/usr/src /usr/src` mount remote filesystem
- `mount server:/usr/src /usr/src` same as above
- `mount -o hard server:/usr/src /usr/src` same as above but hard mount
- `mount -p > /etc/fstab` save current mount state
- `umount -t nfs -b /foo` unmount all nfs filesystems except foo's

**FILES**

- `/etc/fstab` filesystem table
- `/etc/mtab` mount table

**SEE ALSO**

- `fstab(4), mtab(4)`
- `mountd(1M), nfѕd(1M)` if NFS is installed.
- `fsck(1M)`

**NOTE**

If the directory on which a filesystem is to be mounted is a symbolic link, the filesystem is mounted on the directory to which the symbolic link refers, rather than being mounted on top of the symbolic link itself.
NAME
mountall, umountall – mount, unmount multiple file systems

SYNOPSIS
/etc/mountall
/etc/umountall

DESCRIPTION
These commands may be executed only by the super-user.
Before each file system is mounted, it is checked using fsstat(1M) to see if it appears mountable. If the file system does not appear mountable, it is checked, using fsck(1M), before the mount is attempted.

umountall causes all mounted file systems except root to be unmounted.

SEE ALSO
fsck(1M), fsstat(1M), fuser(1M), mount(1M).

DIAGNOSTICS
No messages are printed if the file systems are mountable and clean.
Error and warning messages come from fsck(1M), fsstat(1M), and mount(1M).
NAME

mrouted – IP multicast routing daemon

SYNOPSIS

/usr/etc/mrouted [ -d [ debug_level ] ]

DESCRIPTION

mrouted is an implementation of the Distance-Vector Multicast Routing Protocol (DVMRP), an earlier version of which is specified in RFC-1075. It maintains topological knowledge via a distance-vector routing protocol (like RIP, described in RFC-1058), upon which it implements a multicast forwarding algorithm called Truncated Reverse Path Broadcasting (TRPB).

mrouted forwards a multicast datagram along a shortest (reverse) path tree rooted at the subnet on which the datagram originates. It is a broadcast tree, which means it includes all subnets reachable by a cooperating set of mrouted routers. However, the datagram will not be forwarded onto leaf subnets of the tree if those subnets do not have members of the destination group. Furthermore, the IP time-to-live of a multicast datagram may prevent it from being forwarded along the entire tree.

In order to support multicasting among subnets that are separated by (unicast) routers that do not support IP multicasting, mrouted includes support for "tunnels", which are virtual point-to-point links between pairs of mrouted's located anywhere in an internet. IP multicast packets are encapsulated for transmission through tunnels, so that they look like normal unicast datagrams to intervening routers and subnets. The encapsulation takes the form of an IP source route which is inserted on entry to a tunnel, and stripped out on exit from a tunnel.

The tunnel mechanism allows mrouted to establish a virtual internet, for the purpose of multicasting only, which is independent of the physical internet, and which may span multiple Autonomous Systems. This capability is intended for experimental support of internet multicasting only, pending widespread support for multicast routing by the regular (unicast) routers. mrouted suffers from the well-known scaling problems of any distance-vector routing protocol, and does not (yet) support hierarchical multicast routing or inter-operation with other multicast routing protocols.

mrouted handles multicast routing only; there may or may not be a unicast router running on the same host as mrouted. With the use of tunnels, it is not necessary for mrouted to have access to more than one physical subnet in order to perform multicast forwarding.
INVOCATION
If no "-d" option is given, or if the debug level is specified as 0, mrouted detaches from the invoking terminal. Otherwise, it remains attached to the invoking terminal and responsive to signals from that terminal. If "-d" is given with no argument, the debug level defaults to 2. Regardless of the debug level, mrouted always writes warning and error messages to the system log daemon. Non-zero debug levels have the following effects:

level 1 all syslog'ed messages are also printed to stderr.
level 2 all level 1 messages plus notifications of "significant" events are printed to stderr.
level 3 all level 2 messages plus notifications of all packet arrivals and departures are printed to stderr.

CONFIGURATION
mrouted automatically configures itself to forward on all multicast-capable interfaces, i.e., interfaces that have the IFF_MULTICAST flag set (excluding the loopback "interface"), and it finds other mrouted's directly reachable via those interfaces. To override the default configuration, or to add tunnel links to other mrouteds, configuration commands may be placed in /usr/etc/mrouted.conf. There are two types of configuration command:

phyint <local-addr> [disable] [metric <m>] [threshold <t>]
tunnel <local-addr> <remote-addr> [metric <m>] [threshold <t>]

The phyint command can be used to disable multicast routing on the physical interface identified by local IP address <local-addr>, or to associate a non-default metric or threshold with the specified physical interface. Phyint commands should precede tunnel commands.

The tunnel command can be used to establish a tunnel link between local IP address <local-addr> and remote IP address <remote-addr>, and to associate a non-default metric or threshold with that tunnel. The tunnel must be set up in the mrouted.conf files of both ends before it will be used.

The metric is the "cost" associated with sending a datagram on the given interface or tunnel; it may be used to influence the choice of routes. The metric defaults to 1. Metrics should be kept as small as possible, because mrouted cannot route along paths with a sum of metrics greater than 31.
When in doubt, the following metrics are recommended:

1. LAN, or tunnel across a single LAN
2. serial link, or tunnel across a single serial link
3. multi-hop tunnel

The threshold is the minimum IP time-to-live required for a multicast datagram to be forwarded to the given interface or tunnel. It is used to control the scope of multicast datagrams. (The TTL of forwarded packets is only compared to the threshold, it is not decremented by the threshold. Every multicast router decrements the TTL by 1.) The default threshold is 1. Suggested thresholds:

32 for links that separate sites,
64 for links that separate regions,
128 for links that separate continents.

In general, all mrouteds connected to a particular subnet or tunnel should use the same metric and threshold for that subnet or tunnel.

mrouted will not initiate execution if it has fewer than two enabled vifs, where a vif (virtual interface) is either a physical multicast-capable interface or a tunnel. It will log a warning if all of its vifs are tunnels; such an mrouted configuration would be better replaced by more direct tunnels (i.e., eliminate the middle man).

**Signals**

mrouted responds to the following signals:

HUP
TERM
INT terminates execution gracefully (i.e., by sending good-bye messages to all neighboring routers).
USR1 dumps the internal routing tables to /ust/tmp/mrouted.dump.
QUIT dumps the internal routing tables to stderr (only if mrouted was invoked with a non-zero debug level).
EXAMPLE

The routing tables look like this:

Virtual Interface Table

<table>
<thead>
<tr>
<th>Vif</th>
<th>Local-Address</th>
<th>Metric</th>
<th>Thresh</th>
<th>Flags</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>36.2.0.8</td>
<td>subnet: 36.2</td>
<td>1</td>
<td>querier</td>
</tr>
<tr>
<td></td>
<td></td>
<td>groups: 224.0.2.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>224.0.0.4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>36.11.0.1</td>
<td>subnet: 36.11</td>
<td>1</td>
<td>querier</td>
</tr>
<tr>
<td></td>
<td></td>
<td>groups: 224.0.2.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>224.0.1.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>224.0.0.4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>36.2.0.8</td>
<td>tunnel: 36.8.0.77</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>peers: 36.8.0.77</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>36.2.0.8</td>
<td>tunnel: 36.8.0.110</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

Multicast Routing Table

<table>
<thead>
<tr>
<th>Origin-Subnet</th>
<th>From-Gateway</th>
<th>Metric</th>
<th>In-Vif</th>
<th>Out-Vifs</th>
</tr>
</thead>
<tbody>
<tr>
<td>36.2</td>
<td>36.8</td>
<td>1</td>
<td>0</td>
<td>1* 2 3*</td>
</tr>
<tr>
<td>36.8</td>
<td>36.8.0.77</td>
<td>4</td>
<td>2</td>
<td>0* 1* 3*</td>
</tr>
<tr>
<td>36.11</td>
<td>36.8</td>
<td>1</td>
<td>1</td>
<td>0* 2 3*</td>
</tr>
</tbody>
</table>

In this example, there are four vifs connecting to two subnets and two tunnels. The vif 3 tunnel is not in use (no peer address). The vif 0 and vif 1 subnets have some groups present; tunnels never have any groups. This instance of mrouted is the one responsible for sending periodic group membership queries on the vif 0 and vif 1 subnets, as indicated by the "querier" flags.

Associated with each subnet from which a multicast datagram can originate is the address of the previous hop gateway (unless the subnet is directly-connected), the metric of the path back to the origin, the incoming vif for multicasts from that origin, and a list of outgoing vifs. "*" means that the outgoing vif is connected to a leaf of the broadcast tree rooted at the origin, and a multicast datagram from that origin will be forwarded on that outgoing vif only if there are members of the destination group on that leaf.

FILES

/usr/etc/mrouted.conf

SEE ALSO

TRPB is described, along with other multicast routing algorithms, in the paper "Multicast Routing in Internetworks and Extended LANs" by S. Deering, in the Proceedings of the ACM SIGCOMM '88 Conference.
NAME
multi – switch the system to multi-user mode

SYNOPSIS
/etc/multi

DESCRIPTION
Multi switches the system to multi-user mode if it was in single-user mode
or causes it to re-read its /etc/inittab file and turn the appropriate gettys on
and off. Multi is a shell script that invokes /etc/telin tit.

SEE ALSO
single(1M), init(1M), getty(1M), inittab(4).
NAME
mvdir – move a directory

SYNOPSIS
/etc/mvdir dirname name

DESCRIPTION
mvdir moves directories within a file system. Dirname must be a directory. If name does not exist, it will be created as a directory. If name does exist, dirname will be created as name/dirname. Dirname and name may not be on the same path; that is, one may not be subordinate to the other. For example:

mvdir x/y x/z

is legal, but

mvdir x/y x/y/z

is not.

SEE ALSO

WARNINGS
Only the super-user can use mvdir.
NAME
named – Internet domain name server

SYNOPSIS
/usr/etc/named [-d debuglevel] [-p port#] [(-b) bootfile]

DESCRIPTION
Named is the Internet domain name server. It replaces the original host
Table lookup of information in the network hosts file /etc/hosts. (See
RFC1034 for more information on the Internet name-domain system.)

Named is started at system initialization if the configuration flag named is
set "on" with chkconfig(1M). Without any arguments, named will read the
default boot file /usr/etc/named.d/named.boot, read any initial data and
listen for queries.

Site-dependent options and arguments to named belong in the file
/etc/config/named.options. Options are:

-d Print debugging information. A number after the "d" determines
the level of messages printed.

-p Use a different port number. The default is the standard port
number as listed in /etc/services.

-b Use an alternate boot file. This is optional and allows you to
specify a file with a leading dash.

Any additional argument is taken as the name of the boot file. The boot file
contains information about where the name server is to get its initial data.
If multiple boot files are specified, only the last is used. Lines in the boot
file cannot be continued on subsequent lines. The following is a small
example:

directory /usr/etc/named.d

; type domain source host/file backup file

  cache . . . root.cache
  primary Berkeley.EDU . berkeley.edu.zone
  primary 32.128.IN-ADDR.ARPA . ucbhosts.rev
  secondary CC.Berkeley.EDU 128.32.137.8 128.32.137.3 cc.zone.bak
  secondary 6.32.128.IN-ADDR.ARPA 128.32.137.8 128.32.137.3 cc.rev.bak
  primary 0.0.127.IN-ADDR.ARPA . localhost.rev
  forwarders 10.0.0.78 10.2.0.78

; slave

The "directory" line causes the server to change its working directory to

April 1990 - 1 - Version 5.0
the directory specified. This can be important for the correct processing of \$INCLUDE files in primary zone files.

The "cache" line specifies that data in "root.cache" is to be placed in the backup cache. Its main use is to specify data such as locations of root domain servers. This cache is not used during normal operation, but is used as "hints" to find the current root servers. The file "root.cache" is in the same format as "berkeley.edu.zone". There can be more than one "cache" file specified. The cache files are processed in such a way as to preserve the time-to-live's of data dumped out. Data for the root name servers is kept artificially valid if necessary.

The first "primary" line states that the file "berkeley.edu.zone" contains authoritative data for the "Berkeley.EDU" zone. The file "berkeley.edu.zone" contains data in the master file format described in RFC1034. All domain names are relative to the origin, in this case, "Berkeley.EDU" (see below for a more detailed description). The second "primary" line states that the file "ubchosts.rev" contains authoritative data for the domain "32.128.IN-ADDR.ARPA," which is used to translate addresses in network 128.32 to hostnames. Each master file should begin with an SOA record for the zone (see below).

The first "secondary" line specifies that all authoritative data under "CC.Berkeley.EDU" is to be transferred from the name server at 128.32.137.8. If the transfer fails it will try 128.32.137.3 and continue trying the addresses, up to 10, listed on this line. The secondary copy is also authoritative for the specified domain. The first non-dotted-quad address on this line will be taken as a filename in which to backup the transferred zone. The name server will load the zone from this backup file if it exists when it boots, providing a complete copy even if the master servers are unreachable. Whenever a new copy of the domain is received by automatic zone transfer from one of the master servers, this file will be updated. The second "secondary" line states that the address-to-hostname mapping for the subnet 128.32.136 should be obtained from the same list of master servers as the previous zone.

The "forwarders" line specifies the addresses of sitewide servers that will accept recursive queries from other servers. If the boot file specifies one or more forwarders, then the server will send all queries for data not in the cache to the forwarders first. Each forwarder will be asked in turn until an answer is returned or the list is exhausted. If no answer is forthcoming from a forwarder, the server will continue as it would have without the forwarders line unless it is in "slave" mode. The forwarding facility is useful to cause a large sitewide cache to be generated on a master, and to reduce traffic over links to outside servers. It can also be used to allow servers to run that do not have access directly to the Internet, but wish to act as though
they do.

The "slave" line (shown commented out) is used to put the server in slave mode. In this mode, the server will only make queries to forwarders. This option is normally used on machine that wish to run a server but for physical or administrative reasons cannot be given access to the Internet, but have access to a host that does have access.

The "sortlist" line can be used to indicate networks that are to be preferred over other, unlisted networks. Queries for host addresses from hosts on the same network as the server will receive responses with local network addresses listed first, then addresses on the sort list, then other addresses. This line is only acted on at initial startup. When reloading the name server with a SIGHUP, this line will be ignored.

The master file consists of control information and a list of resource records for objects in the zone of the forms:

$INCLUDE <filename> <opt_domain>
$ORIGIN <domain>
<domain> <opt_ttl> <opt_class> <type> <resource_record_data>

where domain is "." for root, "@" for the current origin, or a standard domain name. If domain is a standard domain name that does not end with "." , the current origin is appended to the domain. Domain names ending with "." are unmodified. The opt_domain field is used to define an origin for the data in an included file. It is equivalent to placing a $ORIGIN statement before the first line of the included file. The field is optional. Neither the opt_domain field nor $ORIGIN statements in the included file modify the current origin for this file. The opt_ttl field is an optional integer number for the time-to-live field. It defaults to zero, meaning the minimum value specified in the SOA record for the zone. The opt_class field is the object address type; currently only one type is supported, IN, for objects connected to the Internet. The type field contains one of the following tokens; the data expected in the resource_record_data field is in parentheses.

A a host address (dotted quad)
NS an authoritative name server (domain)
MX a mail exchanger (domain)
CNAME the canonical name for an alias (domain)
SOA     marks the start of a zone of authority (domain of originating host,  
domain address of maintainer, a serial number and the following  
parameters in seconds: refresh, retry, expire and minimum TTL  
(see RFC1034))
MB      a mailbox domain name (domain)
MG      a mail group member (domain)
MR      a mail rename domain name (domain)
NULL    a null resource record (no format or data)
WKS     a well-known service description
PTR     a domain name pointer (domain)
HINFO   host information (cpu_type OS_type)
MINFO   mailbox or mail list information (request_domain error_domain)

Resource records normally end at the end of a line, but may be continued  
across lines between opening and closing parentheses. Comments are intro-  
duced by semicolons and continue to the end of the line.

Each master zone file should begin with an SOA record for the zone. An  
example SOA record is as follows:
@ IN SOA ucbvax.Berkeley.EDU. rwh.ucbvax.Berkeley.EDU. (  
  2.89    ; serial  
  10800   ; refresh  
  3600    ; retry  
  3600000; expire  
  86400  ) ; minimum

The SOA lists a serial number, which should be changed each time the mas-  
ter file is changed. Secondary servers check the serial number at intervals  
specified by the refresh time in seconds; if the serial number changes, a  
zone transfer will be done to load the new data. If a master server cannot  
be contacted when a refresh is due, the retry time specifies the interval at  
which refreshes should be attempted until successful. If a master server  
cannot be contacted within the interval given by the expire time, all data  
from the zone is discarded by secondary servers. The minimum value is the  
time-to-live used by records in the file with no explicit time-to-live value.

NOTES

The host file directives "domain" and "suffixes" have been obsoleted by  
a more useful resolver based implementation of suffixing for partially  
qualified domain names. The prior mechanisms could fail under a number  
of situations, especially when then local name server did not have complete
information.

The following signals have the specified effect when sent to the server process using the `kill(1)` or `killall(1M)` commands.

**SIGHUP** Causes server to read `named.boot` and reload the database.

**SIGINT** Dumps current data base and cache to `/usr/tmp/named_dump.db`

**SIGABRT** Dumps statistics data into `/usr/tmp/named.stats`. Statistics data is appended to the file.

**SIGUSR1** Turns on debugging; each SIGUSR1 increments debug level.

**SIGUSR2** Turns off debugging completely.

The shell script `/usr/etc/named.reload` sends a SIGHUP to the server. `/usr/etc/named.restart` kills and restarts the server.

**FILES**

- `/usr/etc/named.d/named.boot` name server configuration boot file
- `/usr/tmp/named.run` debug output
- `/usr/tmp/named_dump.db` dump of the name server database
- `/usr/tmp/named.stats` name server statistics data

**SEE ALSO**

`kill(1)`, `gethostname(3N)`, `resolver(3N)`, `hostname(4)`, `resolver(4)`.

*BIND Name Server Operations Guide* in the *Network Communications Guide*.

RFC1032, RFC1033, RFC1034, RFC1035, RFC974
NAME
ncheck – generate path names from i-numbers

SYNOPSIS
/etc/ncheck [ -i i-numbers ] [ -a ] [ -s ] [ file-system ]

DESCRIPTION
ncheck with no arguments generates a path-name vs. i-number list of all files on a set of default file systems (see /etc/fstab). Names of directory files are followed by ./.

The options are as follows:

- i  limits the report to only those files whose i-numbers follow.
- a  allows printing of the names . and .., which are ordinarily suppressed.
- s  limits the report to special files and files with set-user-ID mode. This option may be used to detect violations of security policy.

File system must be specified by the file system’s special file in /dev.

ncheck is only useful with extent file systems.

The report should be sorted so that it is more useful.

SEE ALSO
fsck(1M).

DIAGNOSTICS
If the file system structure is not consistent, ?? denotes the "parent" of a parentless file and a path-name beginning with ... denotes a loop.
NAME

network – network initialization and shutdown script

SYNOPSIS

/etc/init.d/network [ start | stop ]

DESCRIPTION

The network shell script is called during system startup from /etc/rc2 to initialize the standard and optional network devices and daemons. The script is called during system shutdown from /etc/rc0 to gracefully kill the daemons and inactivate the devices.

When called with the start argument, the network script does the following, using the various configuration flags described below:

- Defines the hostname and hostid based on the name in /etc/sys_id and its corresponding Internet address in /etc/hosts.

- Checks that the host’s Internet address is not the default 192.0.2.1 Internet test address. If the address is the default address, the software is configured for standalone mode. An Internet address other than the default must be chosen in order to configure the network properly. See the network administration chapter in the Network Communications Guide for information on selecting an address.

- Initializes the network interfaces. The primary and secondary (if installed) ethernet interfaces are initialized automatically. The Hypernet interface is initialized if the hypernet configuration flag is on. Each interface must have an unique Internet address and name in /etc/hosts. The script derives the names from /etc/sys_id. The prefix gate— is prepended to the host name to generate the second interface’s name. The suffix _hy is appended to generate the Hypernet interface’s name. For example:

  191.50.1.7    yosemite
  191.50.2.49   gate–yosemite
  191.51.0.88   yosemite–hy

If your IRIS-4D has more than 2 ethernet controllers, edit the script to configure the additional interfaces after the standard ones.

- Starts the routing, portmap and named daemons. Initializes the default multicast route.

- Defines the YP domain name, starts the YP and NFS daemons, and mounts NFS filesystems (if NFS is installed).

- Starts the inetd, timed, timeslave and rwhod daemons. Starts the rarpd daemon if diskless support is installed.
• Starts the 4DDN software (if installed).

When called with the stop argument, the network script gracefully terminates daemons in the correct order, unmounts NFS filesystems and inactivates the network interfaces.

CONFIGURATION FLAGS

A daemon or subsystem is enabled if its configuration flag in the /etc/config directory in the “on” state. If a flag file is missing, the flag is considered off. Use the chkconfig (1M) command to turn a flag on or off. For example,

cchkconfig timed on

enables the timed flag. When invoked without arguments, chkconfig prints the state of all known flags.

There are two special flags: verbose and network. The verbose flag controls the printing of the names of daemons as they are started and the printing of NFS-mounted filesystem names as they are mounted and unmounted. The network flag allows incoming and outgoing traffic. This flag can be set on if you need to isolate the machine from network without removing cables.

The following table lists the configuration flags used to initialize standard and optional software.

<table>
<thead>
<tr>
<th>Flag</th>
<th>Action if “on”</th>
</tr>
</thead>
<tbody>
<tr>
<td>gated</td>
<td>Start Cornell Internet super-routing daemon.</td>
</tr>
<tr>
<td>mrouted</td>
<td>Start Stanford IP multicast routing daemon.</td>
</tr>
<tr>
<td>named</td>
<td>Start 4.3BSD Internet domain name server.</td>
</tr>
<tr>
<td>rarpd</td>
<td>Start the Reverse ARP daemon.</td>
</tr>
<tr>
<td>rwhod</td>
<td>Start 4.3BSD rwho daemon.</td>
</tr>
<tr>
<td>timed</td>
<td>Start 4.3BSD time synchronization daemon.</td>
</tr>
<tr>
<td>timeslave</td>
<td>Start SGI time synchronization daemon.</td>
</tr>
<tr>
<td>hyernet</td>
<td>Initialize Hyernet controller and routes.</td>
</tr>
<tr>
<td>nfs</td>
<td>Start NFS daemons, mount NFS filesystems.</td>
</tr>
<tr>
<td>automount</td>
<td>Start NFS automounter daemon.</td>
</tr>
<tr>
<td>yp</td>
<td>Enable Yellow Pages, start ypbind daemon.</td>
</tr>
<tr>
<td>ypserv</td>
<td>If yp is on, become a YP server.</td>
</tr>
<tr>
<td>ypmaster</td>
<td>If yp is on, become the YP master; start passwd server ypserv should be on, too.</td>
</tr>
<tr>
<td>4DDN</td>
<td>Initialize 4DDN (DECnet connectivity) software.</td>
</tr>
</tbody>
</table>

Site-dependent options for daemons belong in “options” files in /etc/config. Certain daemons require options so their options file must contain valid information. See the Network Communications Guide and the
daemon’s manual page in section 1M for details on valid options.

    File                  Status
    automount.options     optional
    gated.options         optional
    ifconfig-1.options    optional (for primary network interface)
    ifconfig-2.options    optional (for gateway network interface)
    ifconfig-hy.options   optional (for Hypernet interface)
    inetd.options        optional
    mroute.options        optional
    named.options         optional
    rarpd.options         optional
    routed.options        optional
    rpc.passwd.options    optional
    rwhod.options         optional
    timed.options         optional
    timeslave.options     required
    ypserv.options        optional

Site-dependent configuration commands to start and stop local daemons, add static routes and publish arp entries should be put in a separate shell script called /etc/init.d/network.local. Make symbolic links in /etc/rc0.d and /etc/rc2.d to this file to have it called during system startup and shutdown:

    ln -s /etc/init.d/network.local /etc/rc0.d/K39network
    ln -s /etc/init.d/network.local /etc/rc2.d/S31network

See /etc/init.d/network for the general format of the script.

FILES

/etc/init.d/network
/etc/rc0.d/K40network linked to network
/etc/rc2.d/S30network linked to network
/etc/config configuration flags and options files
/etc/sys_id host name
/etc/hosts Internet address-name database
/usr/etc/yp/ypdomain YP domain name

SEE ALSO

chkconfig(1M), rc0(1M), rc2(1M)
Network Communications Guide for the IRIS-4D Series.
NAME
newaliases – rebuild the data base for the mail aliases file

SYNOPSIS
newaliases

DESCRIPTION
Newaliases rebuilds the random access data base for the mail aliases file
/usr/lib/aliases. It must be run each time /usr/lib/aliases is changed in order
for the change to take effect, unless sendmail has been configured to
automatically rebuild the database, which is the default.

SEE ALSO
aliases(4), sendmail(1M)
NAME
nvram, sgikopt – get or set non-volatile RAM variable(s)

SYNOPSIS
nvram [-v] [name [value]]
sgikopt [name...]

DESCRIPTION
Nvram may be used to set or print the values of non-volatile RAM variables. If name is specified, nvram prints the corresponding value. If value is specified and name is defined in non-volatile RAM, nvram replaces name’s definition string with value. The -v option causes nvram to print a line of the form name=value after getting or setting the named variable. When invoked with no arguments, all known variables are displayed in the name=value form.

If invoked as sgikopt, more than one name may be given. Names that do not match known variables are ignored. The exit status is 1 if any arguments don’t match, and 0 otherwise.

NOTES
Non-volatile RAM contains a small set of well-known strings at fixed offsets. Nvram may not be used to define new variables.

Only the super-user may set variables.

The term "Non-volatile RAM" is somewhat misleading, because some variables are placed only in volatile RAM, and will be reset on power-up. Different models have different mixes of volatile and non-volatile variables.

DIAGNOSTICS
If an attempt to get or set a variable fails for any reason, nvram prints an appropriate message on standard error and exits with non-zero status.

SEE ALSO
sgikopt(2), syssgi(2)
NAME
   pac – printer/plotter accounting information

SYNOPSIS
   /etc/pac [ -Pprinter ] [ -pprice ] [ -s ] [ -r ] [ -c ] [ -m ] [ name ... ]

DESCRIPTION
   Pac reads the printer/plotter accounting files, accumulating the number of
   pages (the usual case) or feet (for raster devices) of paper consumed by
   each user, and printing out how much each user consumed in pages or feet
   and dollars. If any names are specified, then statistics are only printed for
   those users; usually, statistics are printed for every user who has used any
   paper.

   The -P flag causes accounting to be done for the named printer. Normally,
   accounting is done for the default printer (site dependent) or the value of
   the environment variable PRINTER is used.

   The -p flag causes the value price to be used for the cost in dollars instead
   of the default value of 0.02 or the price specified in /etc/printcap.

   The -c flag causes the output to be sorted by cost; usually the output is
   sorted alphabetically by name.

   The -r flag reverses the sorting order.

   The -s flag causes the accounting information to be summarized on the
   summary accounting file; this summarization is necessary since on a busy
   system, the accounting file can grow by several lines per day.

   The -m flag causes the host name to be ignored in the accounting file. This
   allows for a user on multiple machines to have all of his printing charges
   grouped together.

FILES
   /usr/adm/?acct         raw accounting files
   /usr/adm/?_sum         summary accounting files
   /etc/printcap          printer capability data base

SEE ALSO
   lpc(1), lpr(1), lpq(1), lprm(1), printcap(4)
   BSD Line Printer Spooler Manual

BUGS
   The relationship between the computed price and reality is as yet unknown.
NAME
passmgmt – password file management

SYNOPSIS
passmgmt –a options name
passmgmt –m options name
passmgmt –d name

DESCRIPTION
Passmgmt updates information in the password file /etc/passwd.

passmgmt –a adds an entry for user name to the password file and
passmgmt –a +name adds a Yellow Pages entry for user name to the pass-
word file. This command does not create any directory for the new user
and the new login remains without a password until the passwd(1) com-
mand is executed to set the password.

passmgmt –m modifies the entry for user name in the password file. All
fields except password in the /etc/passwd entry can be modified by this
command. Only fields specified on the command line will be modified.

passmgmt –d deletes the entry for user name from the password file. It
will not remove any files that the user owns on the system; they must be
removed manually.

The following options are available:

–c comment A short description of the login. It is limited to a maximum
of 128 characters and defaults to an empty field.

–h homedir Home directory of name. It is limited to a maximum of 256
characters and defaults to /usr/people/name.

–u uid UID of the name. This number must range from 0 to the
maximum non-negative value for the system. It defaults to
the next available UID greater than 99. For a Yellow Pages
entry, the default is 0. Without the –o option, passmgmt
enforces the uniqueness of a UID.

–o This option allows a UID to be non-unique. It is used only
with the –u option.

–g gid GID of the name. This number must range from 0 to the
maximum non-negative value for the system. The default is
1 for a local entry and 0 for a Yellow Pages entry.

–s shell Login shell for name. It should be the full pathname of
the program that will be executed when the user logs in. The
maximum size of shell is 256 characters. The default is for
this field to be empty and to be interpreted as /bin/sh.
-l logname  This option changes the name to logname. It also can change a local entry to a Yellow Pages entry by passmgmt
-m -l +name name, or change a Yellow Pages entry to a local entry by passmgmt -m -l name +name. It is used only with the -m option.

FILES
/etc/passwd, /etc/opasswd, /etc/.pwd.lock

SEE ALSO
passwd(1), ypchpass(1), yppasswd(1), passwd(4)

DIAGNOSTICS
The passmgmt command exits with one of the following values:
0    SUCCESS.
1    Permission denied.
2    Invalid command syntax. A usage message is displayed.
3    Invalid argument provided to option.
4    UID in use.
6    Unexpected failure. Password files unchanged.
7    Unexpected failure. Password file(s) missing.
8    Password file(s) busy. Try again later.
9    name does not exist (if -m or -d is specified), already exists (if -a is specified), or logname already exists (if -m -l is specified).

NOTE
You cannot use a colon or <cr> as part of an argument because it will be interpreted as a field separator in the password file.
NAME
ping – send ICMP ECHO_REQUEST packets to network hosts

SYNOPSIS

DESCRIPTION
Ping is a tool for network testing, measurement and management. It utilizes the ICMP protocol’s ECHO_REQUEST datagram to elicit an ICMP ECHO_RESPONSE from a host or gateway. ECHO_REQUEST datagrams ("pings") have an IP and ICMP header, followed by an 8-byte timestamp, and then an arbitrary number of "pad" bytes used to fill out the packet.

The host can be the name of a host or its Internet address. The options are:

- c count
  Stop after sending (and receiving) count ECHO_RESPONSE packets.

- d
  Set the SO_DEBUG option on the socket being used.

- f
  Flood ping. Outputs packets as fast as they come back or one hundred times per second, whichever is more. For every ECHO_REQUEST sent a period "." is printed, while for every ECHO_REPLY received a backspace is printed. This provides a rapid display of how many packets are being dropped. This can be extremely stressful on a network and should be used with caution.

- i interval
  Wait interval seconds between sending each packet. The default is to wait for one second between each packet. This option is incompatible with the -f option.

- l preload
  Send preload packets as fast as possible before falling into the normal mode of behavior.

- n
  Numeric output only. No attempt will be made to lookup symbolic names for host addresses. Useful if your name server is flakey or for hosts not in the database.

- p pattern
  You may specify up to 16 "pad" bytes to fill out the packet you send. This is useful for diagnosing data-dependent problems in a network. For example, "-p ff" will cause the sent packet to be filled with all ones.
-q Quiet output. Nothing is displayed except the summary line on termination.

-r Bypass the normal routing tables and send directly to a host on an attached network. If the host is not on a directly-attached network, an error is returned. This option can be used to ping a local host through an interface that has no route through it (e.g., after the interface was dropped by routed(1M)).

-s size Send datagrams containing size bytes of data. The default is 56, which translates into 64 ICMP data bytes when combined with the 8 bytes of ICMP header data. The maximum allowed value is 65468 bytes.

-v Verbose output. ICMP packets other than ECHO RESPONSE that are received are listed.

-I interface Send multicast datagrams on the network interface specified by the interface’s hostname or IP address.

-L When sending to a multicast destination address, don’t loop the datagram back to ourselves.

-R Record Route. Includes the RECORD_ROUTE option in the ECHO_REQUEST packet and displays the route buffer on returned packets. Note that the IP header is only large enough for six such routes. Many hosts ignore or discard this option.

-T ttl Changes the default time-to-live for datagrams sent to a multicast address.

Ping should be used primarily for manual fault isolation. Because of the load it can impose on the network, it is unwise to use ping during normal operations or from automated scripts. When using ping for fault isolation, it should first be run on the local host, to verify that the local network interface is up and running. Then, hosts and gateways further and further away should be “pinged”.

Ping continually sends one datagram per second, and prints one line of output for every ECHO_RESPONSE returned. If the -c count option is given, only that number of requests is sent. No output is produced if there is no response. Round-trip times and packet loss statistics are computed. If duplicate packets are received, they are not included in the packet loss calculation, although the round trip time of these packets is used in calculating the minimum/average/maximum round-trip time numbers. When the specified number of packets have been sent (and received) or if the program is terminated with an interrupt (SIGINT), a brief summary is displayed. When not using the -f (flood) option, the first interrupt, usually generated
by control-C or DEL, causes ping to wait for its outstanding requests to return. It will wait no longer than the longest round trip time encountered by previous, successful pings. The second interrupt stops ping immediately.

DETAILS
An IP header without options in 20 bytes. An ICMP ECHO_REQUEST packet contains an additional 8 bytes worth of ICMP header followed by an arbitrary amount of data. When a packet size is given, this indicated the size of this extra piece of data (the default is 56). Thus the amount of data received inside of an IP packet of type ICMP ECHO_REPLY will always be 8 bytes more than the requested data space (the ICMP header).

If the data space is at least eight bytes large, ping uses the first eight bytes of this space to include a timestamp which it uses in the computation of round trip times. If less than eight bytes of pad are specified, no round trip times are given.

DUPLICATE AND DAMAGED PACKETS
Ping will report duplicate and damaged packets. Duplicate packets should never occur, and seem to be caused by inappropriate link-level retransmissions. Duplicates may occur in many situations and are rarely (if ever) a good sign, although the presence of low levels of duplicates may not always be cause for alarm.

Damaged packets are obviously serious cause for alarm and often indicate broken hardware somewhere in the ping packet’s path (in the network or in the hosts).

TRYING DIFFERENT DATA PATTERNS
The (inter)network layer should never treat packets differently depending on the data contained in the data portion. Unfortunately, data-dependent problems have been known to sneak into networks and remain undetected for long periods of time. In many cases the particular pattern that will have problems is something that doesn’t have sufficient “transitions”, such as all ones or all zeros, or a pattern right at the edge, such as almost all zeros. It isn’t necessarily enough to specify a data pattern of all zeros (for example) on the command line because the pattern that is of interest is at the data link level, and the relationship between what you type and what the controllers transmit can be complicated.

This means that if you have a data-dependent problem you will probably have to do a lot of testing to find it. If you are lucky, you may manage to find a file that either can’t be sent across your network or that takes much longer to transfer than other similar length files. You can then examine this file for repeated patterns that you can test using the –p option of ping.
TTL DETAILS

The TTL value of an IP packet represents the maximum number of IP routers that the packet can go through before being thrown away. In current practice you can expect each router in the Internet to decrement the TTL field by exactly one.

The TCP/IP specification says that the TTL field for TCP packets should be set to 60, but many systems use smaller values (IRIX and 4.3BSD use 30, 4.2BSD used 15).

The maximum possible value of this field is 255, and most Unix systems set the TTL field of ICMP ECHO_REQUEST packets to 255. This is why you will find you can "ping" some hosts, but not reach them with telnet or ftp.

In normal operation ping prints the ttl value from the packet it receives. When a remote system receives a ping packet, it can do one of three things with the TTL field in its response:

• Not change it; this is what Berkeley Unix systems did before the 4.3BSD-tahoe release. In this case the TTL value in the received packet will be 255 minus the number of routers in the round-trip path.

• Set it to 255; this is what IRIX and current Berkeley Unix systems do. In this case the TTL value in the received packet will be 255 minus the number of routers in the path from the remote system to the ping’ing host.

• Set it to some other value. Some machines use the same value for ICMP packets that they use for TCP packets, for example either 30 or 60. Others may use completely wild values.

BUGS

Many Hosts and Gateways ignore the RECORD_ROUTE option.

The maximum IP header length is too small for options like RECORD_ROUTE to be completely useful. There’s not much that that can be done about this, however.

Flood pinging is not recommended in general, and flood pinging the broadcast address should only be done under very controlled conditions.

The record-route option does not work with hosts using network code derived from 4.3BSD.

SEE ALSO
netstat(1), ifconfig(1M), routed(1M)
NAME

portmap – TCP, UDP port to RPC program number mapper

SYNOPSIS

/usr/etc/rpc.portmap

DESCRIPTION

*Portmap* is a server that converts RPC program numbers into TCP or UDP protocol port numbers. It must be running in order to make RPC calls.

When an RPC server is started, it will tell *portmap* what port number it is listening to, and what RPC program numbers it is prepared to serve. When a client wishes to make an RPC call to a given program number, it will first contact *portmap* on the server machine to determine the port number where RPC packets should be sent.

Normally, standard RPC servers are started by *inetd*(1M), so *portmap* must be started before *inetd* is invoked.

SEE ALSO

rpcinfo(1M), inetd(1M)

BUGS

If *portmap* crashes, all servers must be restarted.
POWERDOWN(1M) Silicon Graphics POWERDOWN(1M)

NAME
powerdown – stop all processes and halt the system

SYNOPSIS
powerdown [ -y | -Y ]

DESCRIPTION
This command brings the system to a state where nothing is running so the power can be turned off.

By default, the user is asked questions that control how much warning the other users are given. The options:

- \( y \) prevents the questions from being asked and just gives the warning messages. There is a 60 second pause between the warning messages.

- \( Y \) is the same as \(-y\) except it has no pause between messages. It is the fastest way to bring the system down.

The identical function is also available under the `sysadm` command:

```
sysadm powerdown
```

Password control can be instituted on this command. See `sysadm(1)`, `admpasswd` sub-command.

EXAMPLES

```
some-long-running-command; powerdown -y
```

The first command is run to completion and then the system is halted. This is useful for, say, formatting a document to the printer at the end of a day.

FILES

```
/etc/shutdown - invoked by powerdown
```

SEE ALSO

`shutdown(1M)`, `sysadm(1)` in the *User's Reference Manual*. 

NAME
    preset – reset the lp queue system to a pristine state by deleting printers

SYNOPSIS
    /usr/spool/lp/etc/util/preset

DESCRIPTION
    preset should be used when standard methods of manipulating printers
    (adding, deleting) or the lp printing queue fail. For example, rmprinter
    (1M) can fail if any of the standard lp utilities it calls fail, and the utilities
    can fail if certain files are corrupted. preset deletes and creates crucial files
    in the lp spool directory; after it is run, there are no printers and the system
    will allow printers to be added.

SEE ALSO
    mknptsp(1M), mkPS(1M), mksentpr(1M)
NAME
profiler: prfld, prfstat, prfdc, prfsnap, prfpr – UNIX system profiler

SYNOPSIS
/etc/prfld [ system_namelist ]
/etc/prfstat on
/etc/prfstat off
/etc/prfdc file [ period [ off_hour ] ]
/etc/prfsnap file
/etc/prfpr file [ cutoff [ system_namelist ] ]

DESCRIPTION
Prfld, prfstat, prfdc, prfsnap, and prfpr form a system of programs to facilitate an activity study of the UNIX operating system.

Prfld is used to initialize the recording mechanism in the system. It generates a table containing the starting address of each system subroutine as extracted from system_namelist.

Prfstat is used to enable or disable the sampling mechanism. Profiler overhead is less than 1% as calculated for 500 text addresses. Prfstat will also reveal the number of text addresses being measured.

Prfdc and prfsnap perform the data collection function of the profiler by copying the current value of all the text address counters to a file where the data can be analyzed. Prfdc will store the counters into file every period minutes and will turn off at off_hour (valid values for off_hour are 0–24). Prfsnap collects data at the time of invocation only, appending the counter values to file.

Prfpr formats the data collected by prfdc or prfsnap. Each text address is converted to the nearest text symbol (as found in system_namelist) and is printed if the percent activity for that range is greater than cutoff. cutoff may be given as a floating-point number >= 0.01. If cutoff is zero, then all samples collected are printed, even if their percentage is less than 0.01%.

FILES
/dev/prf interface to profile data and text addresses
/unix default for system namelist file

April 1990

Version 5.0
NAME

prvtoc – print volume header information.

SYNOPSIS

/etc/prvtoc [header_device_name] device

DESCRIPTION

prvtoc prints a summary of the information in the volume header of a disk. (See vh(7m)). The command can be used only by the super-user.

The device name should be the raw device filename of a disk volume header in the form /dev/rdsk/xxx?d?q/vh.

Note: prvtoc knows about the special file directory naming conventions, so the /dev/rdsk prefix may be omitted.

If no name is given, the information for the root disk is printed.

prvtoc prints information about the disk geometry (number of cylinders etc.), followed by information about the partitions. For each partition, the type is indicated (e.g. filesystem, raw data etc.). For filesystem partitions prvtoc shows if there is actually a filesystem on the partition, and if it is mounted, the mount point is shown.

The following options to prvtoc may be used:

-s
Print only the partition table, with headings but without the comments.

-h
Print only the partition table, without headings and comments. Use this option when the output of the prvtoc command is piped into another command.

-tfstab
Use the file fstab instead of /etc/fstab.

-mmntab
Use the file mmntab instead of /etc/mount.
EXAMPLE

The output below is for a 180 megabyte root disk, obtained by invoking `prtvtoc` without parameters.

Printing label for root disk

* /dev/rdsk/ips0d0vh (bootfile "/unix") partition map
* 
* Dimensions:
*  512 bytes/sector
*  32 sectors/track
*  10 tracks/cylinder
*  823 cylinders
*  23 cylinders occupied by header & badblock replacement tracks
*  800 accessible cylinders
* 
* No space unallocated to partitions

<table>
<thead>
<tr>
<th>Partition</th>
<th>Type</th>
<th>Fs</th>
<th>Start: sec (cyl)</th>
<th>Size: sec (cyl)</th>
<th>Mount Directory</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>efs</td>
<td>yes</td>
<td>2240 (7)</td>
<td>32640 (102)</td>
<td>/</td>
</tr>
<tr>
<td>1</td>
<td>raw</td>
<td></td>
<td>34880 (109)</td>
<td>65600 (205)</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>efs</td>
<td>yes</td>
<td>100480 (314)</td>
<td>157760 (493)</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>efs</td>
<td>yes</td>
<td>100480 (314)</td>
<td>157760 (493)</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>efs</td>
<td>yes</td>
<td>100480 (314)</td>
<td>157760 (493)</td>
<td>/usr</td>
</tr>
<tr>
<td>7</td>
<td>efs</td>
<td></td>
<td>2240 (7)</td>
<td>256000 (800)</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>volhdr</td>
<td></td>
<td>0 (0)</td>
<td>2240 (7)</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>trkrep</td>
<td></td>
<td>258240 (807)</td>
<td>5120 (16)</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>volume</td>
<td></td>
<td>0 (0)</td>
<td>263360 (823)</td>
<td></td>
</tr>
</tbody>
</table>

SEE ALSO
dvhtool(1M), fx(1m), vh(7m).
NAME
pwck, grpck - password/group file checkers

SYNOPSIS
/etc/pwck [file]
/etc/grpck [file]

DESCRIPTION
pwck scans the password file and notes any inconsistencies. The checks
include validation of the number of fields, login name, user ID, group ID,
and whether the login directory and the program-to-use-as-Shell exist. The
default password file is /etc/passwd.

Grpck verifies all entries in the group file. This verification includes a check
of the number of fields, group name, group ID, and whether all login names
appear in the password file. The default group file is /etc/group.

FILES
/etc/group
/etc/passwd

SEE ALSO

DIAGNOSTICS
Group entries in /etc/group with no login names are flagged.
NAME
rc0 — run commands performed to stop the operating system

SYNOPSIS
/etc/rc0

DESCRIPTION
This file is executed at each system state change that needs to have the sys-
tem in an inactive state. It is responsible for those actions that bring the
system to a quiescent state, traditionally called “shutdown”.

There are three system states that require this procedure. They are state 0
(the system halt state), state 5 (the firmware state), and state 6 (the reboot
state). Whenever a change to one of these states occurs, the /etc/rc0 pro-
cedure is run. The entry in /etc/inittab might read:

s0:056:wait:/etc/rc0 >/dev/console 2>&1 </dev/console

Some of the actions performed by /etc/rc0 are carried out by files in the
directory /etc/shutdown.d. and files beginning with K in /etc/rc0.d. These
files are executed in ascii order (see FILES below for more information),
terminating some system service. The combination of commands in
/etc/rc0 and files in /etc/shutdown.d and /etc/rc0.d determines how the sys-
tem is shut down.

The recommended sequence for /etc/rc0 is:
Stop System Services and Daemons.

Various system services (such as 3BNET Local Area Network or
LP Spooler) are gracefully terminated.

When new services are added that should be terminated when the
system is shut down, the appropriate files are installed in
/etc/shutdown.d and /etc/rc0.d.

Terminate Processes

SIGTERM signals are sent to all running processes by killall(1M).
Processes stop themselves cleanly if sent SIGTERM.

Kill Processes

SIGKILL signals are sent to all remaining processes; no process
can resist SIGKILL.

At this point the only processes left are those associated with
/etc/rc0 and processes 0 and 1, which are special to the operating
system.

Unmount All File Systems

Only the root file system (/) remains mounted.

Depending on which system state the systems end up in (0, 5, or 6), the entries in /etc/inittab will direct what happens next. If the /etc/inittab has not defined any other actions to be performed as in the case of system state 0, then the operating system will have nothing to do. It should not be possible to get the system’s attention. The only thing that can be done is to turn off the power or possibly get the attention of a firmware monitor. The command can be used only by the super-user.

FILES

The execution by /bin/sh of any files in /etc/shutdown.d occurs in ascii sort-sequence order. See rc2(1M) for more information.

SEE ALSO

killall(1M), rc2(1M), shutdown(1M).
NAME
rc2 – run commands performed for multi-user environment

SYNOPSIS
/etc/rc2

DESCRIPTION
This file is executed via an entry in /etc/inittab and is responsible for those initializations that bring the system to a ready-to-use state, traditionally state 2, called the "multi-user" state.

/etc/rc2 runs files beginning with S in /etc/rc2.d. These files are executed by /bin/sh in ascii sort–sequence order (see FILES below for more information).

Each of these files may also check the state of the corresponding chkconfig flag for that function. If the state is on the script will go on to start that function, if the state is off it will not start that function. (See chkconfig(1m)).

The functions performed by the /etc/rc2 command and associated /etc/rc2.d files include:

   Setting and exporting the TIMEZONE variable.
   Setting-up and mounting the user (/usr) file system.
   Cleaning up (remaking) the /tmp and /usr/tmp directories.
   Initializing the network interfaces, mounting network file systems, and starting the appropriate daemon processes.
   Starting the cron daemon by executing /etc/cron.
   Cleaning up (deleting) uucp locks status, and temporary files in the /usr/spool/uucp directory.

Other functions can be added, as required, to support the addition of hardware and software features.

EXAMPLES
The following are prototypical files found in /etc/rc2.d. These files are prefixed by an S and a number indicating the execution order of the files.

MOUNTFILESYS
   # Set up and mount file systems
   
   cd /
/etc/mountall /etc/fstab

RMTMPFILES
    # clean up /tmp
    rm -rf /tmp
    mkdir /tmp
    chmod 777 /tmp
    chgrp sys /tmp
    chown sys /tmp

uucp
    # clean-up uucp locks, status, and temporary files
    rm -rf /usr/spool/locks/*

The file /etc/TIMEZONE is included early in /etc/rc2, thus establishing the default time zone for all commands that follow.

See rc0(1M) for the system shutdown procedure.

FILES
    Files in /etc/rc2.d must begin with an S or a K followed by a number and the rest of the file name. Upon entering run level 2, files beginning with S are executed with the start option; files beginning with K, are executed with the stop option. Files in /etc/rc2.d are typically symbolic links to files in /etc/init.d. Files beginning with other characters are ignored.

SEE ALSO
    rc0(1M), shutdown(1M).
NAME
reboot — reboot the system

SYNOPSIS
   cd /
   /etc/reboot

DESCRIPTION
   When UNIX is running reboot halts and then restarts the system in an orderly fashion. It is useful after changing the configuration of the system. You must have super-user privilege to use this command. If you are remotely logged-in to the system, you will be prompted to confirm the reboot.

   To halt the system before turning it off, use halt(1M) or shutdown(1M).

SEE ALSO
   init(1M).
NAME
renice – alter priority of running processes

SYNOPSIS
/etc/renice priority [ [ -p ] pid ... ] [ [ -g ] pgrp ... ] [ [ -u ] user ... ]

DESCRIPTION
renice alters the scheduling priority of one or more running processes. renice’ing a process group causes all processes in the process group to have their scheduling priority altered. renice’ing a user causes all processes owned by the user to have their scheduling priority altered.

The parameters are interpreted as process ID’s, process group ID’s, or user names as follows. By default, the processes to be affected are specified by their process ID’s. To force parameters to be interpreted as process group ID’s, a -g may be specified. To force the parameters to be interpreted as user names, a -u may be given. Supplying -p will reset interpretation to be (the default) process ID’s. For example,

/etc/renice +1 987 -u daemon root -p 32

would change the priority of process ID’s 987 and 32, and all processes owned by users daemon and root.

Users other than the super-user may only alter the priority of processes they own, and can only monotonically increase their “nice value” within the range 0 to PRIOR_MAX (20). (This prevents overriding administrative fleas.) The super-user may alter the priority of any process and set the priority to any value in the range PRIOR_MIN (−20) to PRIOR_MAX. Useful priorities are: 20 (the affected processes will run only when nothing else in the system wants to), 0 (the “base” scheduling priority), anything negative (to make things go very fast).

FILES
/etc/passwd to map user names to user ID’s

SEE ALSO
getpriority(2), setpriority(2)
NAME
rexecd – remote execution server

SYNOPSIS
/usr/etc/rexecd

DESCRIPTION

*rexecd* is the server for the *rexc* (3N) routine. The server provides remote execution facilities with authentication based on user names and passwords.

*rexecd* listens for service requests at the port indicated in the "exec" service specification; see *services*(4). When a service request is received the following protocol is initiated:

1) The server reads characters from the socket up to a null ('\0') byte. The resultant string is interpreted as an ASCII number, base 10.

2) If the number received in step 1 is non-zero, it is interpreted as the port number of a secondary stream to be used for the *stderr*. A second connection is then created to the specified port on the client’s machine.

3) A null-terminated user name of at most 16 characters is retrieved on the initial socket.

4) A null-terminated, unencrypted password of at most 16 characters is retrieved on the initial socket.

5) A null-terminated command to be passed to a shell is retrieved on the initial socket. The length of the command is limited by the upper bound on the size of the system’s argument list.

6) *rexecd* then validates the user as is done at login time and, if the authentication was successful, changes to the user’s home directory, and establishes the user and group protections of the user. If any of these steps fail the connection is aborted with a diagnostic message returned.

7) A null byte is returned on the initial socket and the command line is passed to the normal login shell of the user. The shell inherits the network connections established by *rexecd*.

DIAGNOSTICS

Except for the last one listed below, all diagnostic messages are returned on the initial socket, after which any network connections are closed. An error is indicated by a leading byte with a value of 1 (0 is returned in step 7 above upon successful completion of all the steps prior to the command execution).
"username too long"
The name is longer than 16 characters.

"password too long"
The password is longer than 16 characters.

"command too long"
The command line passed exceeds the size of the argument list (as configured into the system).

"Login incorrect."
No password file entry for the user name existed. (Also logged to the syslogd(1M) daemon as an auth.notice message.)

"Password incorrect."
The wrong was password supplied. (Also logged to the syslogd(1M) daemon as an auth.notice message.)

"No remote directory."
The chdir command to the home directory failed.

"Try again."
A fork by the server failed.

"<shellname>: ...
The user's login shell could not be started. This message is returned on the connection associated with the stderr, and is not preceded by a flag byte.

SEE ALSO
reexec(3N)

BUGS
Indicating "Login incorrect" as opposed to "Password incorrect" is a security breach which allows people to probe a system for users with null passwords.

A facility to allow all data and password exchanges to be encrypted should be present.
NAME
rlogind – remote login server

SYNOPSIS
/usr/etc/rlogind

DESCRIPTION
Rlogind is the server for the rlogin(1C) program. The server provides a remote login facility with authentication based on privileged port numbers from trusted hosts.

Rlogind listens for service requests at the port indicated in the "login" service specification; see services(4). When a service request is received the following protocol is initiated:

1) The server checks the client’s source port. If the port is not in the range 513-1023, the server aborts the connection.

2) The server checks the client’s source address and requests the corresponding host name (see gethostbyaddr(3N), hosts(4) and named(1M)). If the hostname cannot be determined, the dot-notation representation of the host address is used.

Once the source port and address have been checked, rlogind allocates a pseudo terminal (see pty(7M)), and manipulates file descriptors so that the slave half of the pseudo terminal becomes the stdin, stdout, and stderr for a login process. The login process is an instance of the login(1) program, invoked with the -r option. The login process then proceeds with the authentication process as described in rshd(1M), but if automatic authentication fails, it reprompts the user to login as one finds on a standard terminal line.

The parent of the login process manipulates the master side of the pseudo terminal, operating as an intermediary between the login process and the client instance of the rlogin program. In normal operation, the packet protocol described in pty(7M) is invoked to provide ^S/^Q type facilities and propagate interrupt signals to the remote programs. The login process propagates the client terminal’s baud rate and terminal type, as found in the environment variable, "TERM"; see environ(5).

DIAGNOSTICS
All diagnostic messages are returned on the connection associated with the stderr, after which any network connections are closed. An error is indicated by a leading byte with a value of 1.

"Try again.”
A fork by the server failed.
"/bin/sh: ..."
The user's login shell could not be started.

BUGS
The authentication procedure used here assumes the integrity of each client machine and the connecting medium. This is insecure, but is useful in an "open" environment.

A facility to allow all data exchanges to be encrypted should be present.
A more extensible protocol should be used.
NAME
rmail – receive mail via UUCP

SYNOPSIS
rmail [ –T ] destination

DESCRIPTION
Rmail combines the 'From ... remove from ...' lines that UUCP adds to the beginning of mail messages into a single From line. It passes the result to sendmail(1M).

Rmail is intended to be executed by uux. It is not intended to be used anything else.

The –T flag is used for debugging, and logs the command used to start sendmail in the system log.

SEE ALSO
mail_att(1), mail_bsd(1), sendmail(1M), syslog(1B), uux(1C)
NAME
rmprinter – remove a printer from the LP spooling system

SYNOPSIS
rmprinter printer

DESCRIPTION
rmprinter performs tasks that undo the actions of mkPS, mkcentpr, and
mknetpr. It removes the interface file, the log file, and calls various
lpadmin(1M) functions to remove references to the printer in the spooler.

FILES
/usr/spool/lp/etc/log/<printer>-log /usr/spool/lp/interface/<printer>

SEE ALSO
lpshut(1M), lp(1), lpadmin(1M), lpsched(1M), lpshut(1M), lpstat(1),
mkcentpr(1M), mknetpr(1M), mkPS(1M).
NAME
rmt — remote magstripe protocol module

SYNOPSIS
/etc/rmt

DESCRIPTION
Rmt is a program used by the remote programs in manipulating a magnetic
tape drive through an interprocess communication connection. Rmt is nor-
mally started up with an rexec(3N) or rcmd(3N) call.

The rmt program accepts requests specific to the manipulation of magnetic
tapes, performs the commands, then responds with a status indication. All
responses are in ASCII and in one of two forms. Successful commands
have responses of

A number\n
where number is an ASCII representation of a decimal number. Unsuccess-
ful commands are responded to with

E error-number\nerror-message\n
where error-number is one of the possible error numbers described in
intro(2) and error-message is the corresponding error string as printed from
a call to perror(3). The protocol is comprised of the following commands
(a space is present between each token).

O device\nmode

Open the specified device using the indicated mode. Device is a full path-
name and mode is an ASCII representation of a decimal number suitable for passing
to open(2). If a device had already been opened, it is closed before a new open is performed.

V version\n
This command is sent by the client program to indicate
the version# of the 'librmt' library that the client pro-
gram is linked with. If rmt own protocol is the same or
more advanced than that of the client program, rmt will
adjust to the client program protocol and return the client
version#. However, if the client program version# is
more advanced than rmt own protocol version number
then rmt will return its actual version number and expect
the client program to adjust to rmt protocol. The
returned value is the ASCII representation of the version
number.
Cdevice
Close the currently open device. The device specified is ignored.

Lwhence\noffset
Perform an lseek(2) operation using the specified parameters. The response value is that returned from the lseek call.

Wcount\nWrite data onto the open device. Rmt reads count bytes from the connection, aborting if a premature end-of-file is encountered. The response value is that returned from the write(2) call.

Rcount
Read count bytes of data from the open device. If count exceeds the size of the data buffer (10 kilobytes), it is truncated to the data buffer size. Rmt then performs the requested read(2) and responds with Acount-read if the read was successful; otherwise an error in the standard format is returned. If the read was successful, the data read is then sent.

Ioperation\ncount
Perform a MTIOCOP ioctl(2) command using the specified parameters. The parameters are interpreted as the ASCII representations of the decimal values to place in the mt_op and mt_count fields of the structure used in the ioctl call. The return value is the count parameter when the operation is successful.

S\nReturn the status of the open device, as obtained with a MTIOCGET ioctl call. If the operation was successful, an "ack" is sent with the size of the status buffer, then the status buffer is sent (in binary).

Q\nPerform a MTSCSIINQ ioctl(2) command. If the operation was successful, an "ack" is sent with the size of the inquiry buffer, then the inquiry buffer is sent (in binary).

B\nPerform a MTIOCGETBLKSIZE ioctl(2) command. If the operation was successful, an "ack" is sent with the size of the block size buffer, then the block size buffer is sent (in binary).

Z\nPerform a fstat(2) system call on the currently opened device. If the operation was successful, an "ack" is sent with the size of the "stat" structure, then the actual "stat" structure is sent (in binary).
Any other command causes *rmt* to exit.

**DIAGNOSTICS**

All responses are of the form described above. If *rmt* is invoked with an argument, that argument will be treated as a file name and debug informations will be logged in that file.

**SEE ALSO**

rcmd(3N), rexec(3N), mtio(7),

**BUGS**

People tempted to use this for a remote file access protocol are discouraged.
NAME
route – manually manipulate the routing tables

SYNOPSIS
/usr/etc/route [ -f ] [ -n ] [ command args ]

DESCRIPTION
Route is a program used to manually manipulate the network routing tables. It normally is not needed, as the system routing table management daemon, routed(1M), should tend to this task.

Route accepts two commands: add, to add a route, and delete, to delete a route.

All commands have the following syntax:

/usr/etc/route command [ net ] destination gateway [ metric ]

where destination is the destination host or network, gateway is the next-hop gateway to which packets should be addressed, and metric is a count indicating the number of hops to the destination. The metric is required for add commands; it must be zero if the destination is on a directly-attached network, and nonzero if the route utilizes one or more gateways. If adding a route with metric 0, the gateway given is the address of this host on the common network, indicating the interface to be used for transmission. Routes to a particular host are distinguished from those to a network by interpreting the Internet address associated with destination. The optional keywords net and host force the destination to be interpreted as a network or a host, respectively. Otherwise, if the destination has a “local address part” of INADDR_ANY, or if the destination is the symbolic name of a network, then the route is assumed to be to a network; otherwise, it is presumed to be a route to a host. If the route is to a destination connected via a gateway, the metric should be greater than 0. All symbolic names specified for a destination or gateway are looked up first as a host name using gethostbyname(3N). If this lookup fails, getnetbyname(3N) is then used to interpret the name as that of a network.

Route uses a raw socket and the SIOCADDRT and SIOCDELRRT ioctl’s to do its work. As such, only the super-user may modify the routing tables.

If the -f option is specified, route will “flush” the routing tables of all gateway entries. If this is used in conjunction with one of the commands described above, the tables are flushed prior to the command’s application.

The -n option prevents attempts to print host and network names symbolically when reporting actions.
DIAGNOSTICS

```
``add [ host | network ] %s: gateway %s flags %x``

The specified route is being added to the tables. The values printed are from the routing table entry supplied in the ioctl call. If the gateway address used was not the primary address of the gateway (the first one returned by gethostbyname), the gateway address is printed numerically as well as symbolically.

```
``delete [ host | network ] %s: gateway %s flags %x``

As above, but when deleting an entry.

```
``%s %s done``

When the -f flag is specified, each routing table entry deleted is indicated with a message of this form.

```
``Network is unreachable``

An attempt to add a route failed because the gateway listed was not on a directly-connected network. The next-hop gateway must be given.

```
``not in table``

A delete operation was attempted for an entry which wasn’t present in the tables.

```
``routing table overflow``

An add operation was attempted, but the system was low on resources and was unable to allocate memory to create the new entry.

SEE ALSO

routed(1M)
NAME
  routed – network routing daemon

SYNOPSIS
  /usr/etc/routed [ -d ] [ -g ] [ -s ] [ -q ] [ -t ] [ logfile ]

DESCRIPTION
  Routed is invoked at boot time to manage the network routing tables. The
  routing daemon uses a variant of the Xerox NS Routing Information Proto-
  col in maintaining up to date kernel routing table entries. It used a general-
  ized protocol capable of use with multiple address types, but is currently
  used only for Internet routing within a cluster of networks.

  In normal operation routed listens on the udp(7P) socket for the route ser-
  vice (see services(4)) for routing information packets. If the host is an
  internetwork router, it periodically supplies copies of its routing tables to
  any directly connected hosts and networks.

  When routed is started, it uses the SIOCGIFCONF ioctl to find those
directly connected interfaces configured into the system and marked “up”
(the software loopback interface is ignored). If multiple interfaces are
present, it is assumed that the host will forward packets between networks.
Routed then transmits a request packet on each interface (using a broadcast
packet if the interface supports it) and enters a loop, listening for request
and response packets from other hosts.

  When a request packet is received, routed formulates a reply based on the
information maintained in its internal tables. The response packet gen-
erated contains a list of known routes, each marked with a “hop count”
metric (a count of 16, or greater, is considered “infinite”). The metric
associated with each route returned provides a metric relative to the sender.
Response packets received by routed are used to update the routing tables if
one of the following conditions is satisfied:

  (1)  No routing table entry exists for the destination network or host,
       and the metric indicates the destination is “reachable” (i.e. the
       hop count is not infinite).

  (2)  The source host of the packet is the same as the router in the exist-
       ing routing table entry. That is, updated information is being
       received from the very internetwork router through which packets
       for the destination are being routed.

  (3)  The existing entry in the routing table has not been updated for
       some time (defined to be 90 seconds) and the route is at least as
cost effective as the current route.
The new route describes a shorter route to the destination than the one currently stored in the routing tables; the metric of the new route is compared against the one stored in the table to decide this.

When an update is applied, routed records the change in its internal tables and updates the kernel routing table. The change is reflected in the next response packet sent.

In addition to processing incoming packets, routed also periodically checks the routing table entries. If an entry has not been updated for 3 minutes, the entry’s metric is set to infinity and marked for deletion. Deletions are delayed an additional 60 seconds to insure the invalidation is propagated throughout the local internet.

Hosts acting as internetwork routers gratuitously supply their routing tables every 30 seconds to all directly connected hosts and networks. The response is sent to the broadcast address on nets capable of that function, to the destination address on point-to-point links, and to the router’s own address on other networks. The normal routing tables are bypassed when sending gratuitous responses. The reception of responses on each network is used to determine that the network and interface are functioning correctly. If no response is received on an interface, another route may be chosen to route around the interface, or the route may be dropped if no alternative is available.

Routed is started during system initialization from /etc/init.d/network using site-dependent options and arguments in the file /etc/config/routed.options. The options are:

- **-d**  Do not run in the background and enable additional debugging information to be logged, such as bad packets received. This option is meant for interactive use — do not put it in the routed.options file.

- **-g**  This flag is used on internetwork routers to offer a route to the “default” destination. This is typically used on a gateway to the Internet, or on a gateway that uses another routing protocol whose routes are not reported to other local routers.

- **-s**  Supplying this option forces routed to supply routing information whether it is acting as an internetwork router or not. This is the default if multiple network interfaces are present, or if a point-to-point link is in use.

- **-q**  This is the opposite of the -s option.
-t If the -t option is specified, all packets sent or received are printed on the standard output. In addition, routed will not divorce itself from the controlling terminal so that interrupts from the keyboard will kill the process.

Any other argument supplied is interpreted as the name of file in which routed's actions should be logged. This log contains information about any changes to the routing tables and, if not tracing all packets, a history of recent messages sent and received which are related to the changed route.

In addition to the facilities described above, routed supports the notion of "distant" passive and active gateways. When routed is started up, it reads the file /etc/gateways to find gateways which may not be located using only information from the SIOGIFCONF ioctl. Gateways specified in this manner should be marked passive if they are not expected to exchange routing information, while gateways marked active should be willing to exchange routing information (i.e. they should have a routed process running on the machine). Routes through passive gateways are installed in the kernel's routing tables once upon startup. Such routes are not included in any routing information transmitted. Active gateways are treated equally to network interfaces. Routing information is distributed to the gateway and if no routing information is received for a period of the time, the associated route is deleted. Gateways marked external are also passive, but are not placed in the kernel routing table nor are they included in routing updates. The function of external entries is to inform routed that another routing process will install such a route, and that alternate routes to that destination should not be installed. Such entries are only required when both routers may learn of routes to the same destination.

The /etc/gateways is comprised of a series of lines, each in the following format:

<net | host> name1 gateway name2 metric value <passive | active | external>

The net or host keyword indicates if the route is to a network or specific host.

Name1 is the name of the destination network or host. This may be a symbolic name located in /etc/networks or /etc/hosts or an Internet address specified in "dot" notation; see inet(3N).

Name2 is the name or address of the gateway to which messages should be forwarded.

Value is a metric indicating the hop count to the destination host or network.
One of the keywords passive, active or external indicates if the gateway should be treated as passive or active (as described above), or whether the gateway is external to the scope of the routed protocol.

Gateways directly attached to the Internet should run gated(1M) in order to use the Exterior Gateway Protocol (EGP) to gather routing information rather than using a static routing table of passive gateways. EGP is required in order to provide routes for local networks to the rest of the Internet system. See gated(1M) for details.

FILES
/etc/gateways for distant gateways
/etc/config/routed.options Site-dependent options

SEE ALSO
gated(1M), udp(7P), icmp(7P)
"Internet Transport Protocols", XSIS 028112, Xerox System Integration Standard.

BUGS
The kernel’s routing tables may not correspond to those of routed when redirects change or add routes. Routed should note any redirects received by reading the ICMP packets received via a raw socket.

Routed should incorporate other routing protocols, such as Xerox NS and EGP. Using separate processes for each requires configuration options to avoid redundant or competing routes.

Routed should listen to intelligent interfaces, and to error protocols, such as ICMP, to gather more information. It does not always detect unidirectional failures in network interfaces (e.g., when the output side fails).
NAME
rpcinfo – report RPC information

SYNOPSIS
/usr/etc/rpcinfo -b program-number version-number
/usr/etc/rpcinfo -d program-number version-number
/usr/etc/rpcinfo -p [ host ]
/usr/etc/rpcinfo -u host program-number [ version-number ]
/usr/etc/rpcinfo -t host program-number [ version-number ]

DESCRIPTION
rpcinfo makes an RPC call to an RPC server and reports what it finds.

OPTIONS
-b Make an RPC broadcast to procedure 0 of the specified program-number and version-number using UDP and report all hosts that respond.
-d Delete the existing registration for the RPC service of the specified program-number and version-number.
-p Probe the portmapper on host, and print a list of all registered RPC programs. If host is not specified, it defaults to the value returned by hostname(1).
-u Make an RPC call to procedure 0 of program-number using UDP, and report whether a response was received. The version-number defaults to 1.
-t Make an RPC call to procedure 0 of program-number using TCP, and report whether a response was received. The version-number defaults to 1.

The program-number argument can be either a name or a number.

FILES
/etc/rpc names for rpc program numbers

SEE ALSO
portmap(1M), rpc(4), RPC Programming Guide in the Network Communications Guide.
NAME
rshd – remote shell server

SYNOPSIS
/usr/etc/rshd [-a][nL]

DESCRIPTION
Rshd is the server for the rcmd(3N) routine and, consequently, for the
rsh(1C) program. The server provides remote execution facilities with
authentication based on privileged port numbers from trusted hosts. The −a
option verifies the remote host name and address match on all incoming
connections. Normally this check is performed only for connections from
hosts in the local domain. The −l option disables validation using .rhosts
files. Transport-level keep-alive messages are enabled unless the −n option
is present. The use of keep-alive messages allows sessions to be timed out
if the client crashes or becomes unreachable. The −L option causes all suc-
cessful accesses to be logged to syslogd(1M) as auth.info messages. These
options should specified in the /usr/etc/inetd.conf file (see inetd(1M)).

Rshd listens for service requests at the port indicated in the “cmd” service
specification; see services(4). When a service request is received the fol-
lowing protocol is initiated:

1) The server checks the client’s source port. If the port is not in the
   range 512–1023, the server aborts the connection.

2) The server reads characters from the socket up to a null ("\0") byte. The
   resultant string is interpreted as an ASCII number, base 10.

3) If the number received in step 2 is non-zero, it is interpreted as the port
   number of a secondary stream to be used for the stderr. A second
   connection is then created to the specified port on the client’s machine. The
   source port of this second connection is in the range 513–1023.

4) The server checks the client’s source address and requests the
corresponding host name (see gethostbyaddr(3N), hosts(4) and
named(1M)). If the hostname cannot be determined, the dot-notation
representation of the host address is used.

5) A null terminated user name of at most 16 characters is retrieved on
   the initial socket. This user name is interpreted as the user identity on
   the client’s machine.

6) A null terminated user name of at most 16 characters is retrieved on
   the initial socket. This user name is interpreted as a user identity to
   use on the server’s machine.
7) A null terminated command to be passed to a shell is retrieved on the initial socket. The length of the command is limited by the upper bound on the size of the system's argument list.

8) Rshd then validates the user according to the following steps. The local (server-end) user name is looked up in the password file. If the lookup fails, the connection is terminated. Rshd then tries to validate the user using ruserok(3N), which uses the file /etc/hosts.equiv and the .rhosts file found in the user’s home directory. If the user is not the super-user, (user id 0), the file /etc/hosts.equiv is consulted for a list of hosts considered “equivalent”. If the client’s host name is present in this file, the authentication is considered successful. If the lookup fails, or the user is the super-user, then the file .rhosts in the home directory of the remote user is checked for the machine name and identity of the user on the client’s machine. If this lookup fails, the connection is terminated. The –I option prevents ruserok(3N) from doing any validation based on the user’s “.rhosts” file, unless the user is the superuser.

9) If the file /etc/nologin exists and the user is not the super-user, the connection is closed.

10) A null byte is returned on the initial socket and the command line is passed to the normal login shell of the user. The shell inherits the network connections established by rshd.

**DIAGNOSTICS**

Except for the last one listed below, all diagnostic messages are returned on the initial socket, after which any network connections are closed. An error is indicated by a leading byte with a value of 1 (0 is returned in step 10 above upon successful completion of all the steps prior to the execution of the login shell).

“locuser too long”
The name of the user on the client’s machine is longer than 16 characters.

“remuser too long”
The name of the user on the remote machine is longer than 16 characters.

“command too long”
The command line passed exceeds the size of the argument list (as configured into the system).

“Login incorrect.”
No password file entry for the user name existed. (Logged to the syslogd(1M) daemon as an auth.notice message.)
"No remote directory."
The chdir command to the home directory failed. (Logged as an auth.notice message.)

"Permission denied."
The authentication procedure described above failed. (Logged as an auth.notice message.)

"Connection received using IP options (ignored)"
The remote host tried to use explicit IP source routing.

"Connection from <host> on illegal port"
The remote host used a nonprivileged port.

"Can't find name for <address>"
No hostname was found for the IP address. The authentication procedure described above will use the IP address.

"Host addr <x> not listed for host <y>"
The remote host's name and address did not match. The authentication procedure described above will use the IP address instead of the name.

"Can't make pipe."
The pipe needed for the stderr, wasn't created.

"Try again."
A fork by the server failed.

"<shellname>: ...
"The user's login shell could not be started. This message is returned on the connection associated with the stderr, and is not preceded by a flag byte.

SEE ALSO
rsh(1C), rcmd(3N), ruserok(3N)

BUGS
The authentication procedure used here assumes the integrity of each client machine and the connecting medium. This is insecure, but is useful in an "open" environment.
A facility to allow all data exchanges to be encrypted should be present.
A more extensible protocol should be used.
NAME
runacct – run daily accounting

SYNOPSIS
/usr/lib/acct/runacct [mmdd [state]]

DESCRIPTION
runacct is the main daily accounting shell procedure. It is normally ini-
tiated via cron(1M). runacct processes connect, fee, disk, and process
accounting files. It also prepares summary files for prdaily or billing pur-
poses. runacct is distributed only to source code licensees.

runacct takes care not to damage active accounting files or summary files in
the event of errors. It records its progress by writing descriptive diagnostic
messages into active. When an error is detected, a message is written to
/dev/console, mail (see mail(1)) is sent to root and adm, and runacct ter-
minates. runacct uses a series of lock files to protect against re-invocation.
The files lock and lock1 are used to prevent simultaneous invocation, and
lastdate is used to prevent more than one invocation per day.

runacct breaks its processing into separate, restartable states using statefile
to remember the last state completed. It accomplishes this by writing the
state name into statefile. runacct then looks in statefile to see what it has
done and to determine what to process next. states are executed in the fol-
lowing order:

- SET UP Move active accounting files into working files.
- WTMP FIX Verify integrity of wtmp file, correcting date
  changes if necessary.
- CONNECT1 Produce connect session records in ctmp.h format.
- CONNECT2 Convert ctmp.h records into tacct.h format.
- PROCESS Convert process accounting records into tacct.h
  format.
- MERGE Merge the connect and process accounting
  records.
- FEES Convert output of chargefee into tacct.h format
  and merge with connect and process accounting
  records.
- DISK Merge disk accounting records with connect, pro-
  cess, and fee accounting records.
MERGETACCT

Merge the daily total accounting records in daytacct with the summary total accounting records in /usr/adm/acct/sum/tacct.

CMS

Produce command summaries.

USEREXIT

Any installation-dependent accounting programs can be included here.

CLEANUP

Cleanup temporary files and exit.

To restart runacct after a failure, first check the active file for diagnostics, then fix up any corrupted data files such as pacct or wtmp. The lock files and lastdate file must be removed before runacct can be restarted. The argument mmdd is necessary if runacct is being restarted, and specifies the month and day for which runacct will rerun the accounting. Entry point for processing is based on the contents of statefile; to override this, include the desired state on the command line to designate where processing should begin.

EXAMPLES

To start runacct.

    nohup runacct 2> /usr/adm/acct/nite/fd2log &

To restart runacct.

    nohup runacct 0601 2>> /usr/adm/acct/nite/fd2log &

To restart runacct at a specific state.

    nohup runacct 0601 MERGE 2>> /usr/adm/acct/nite/fd2log &

FILES

/etc/wtmp
/usr/adm/pacct*
/usr/src/cmd/acct/tacct.h
/usr/src/cmd/acct/ctmp.h
/usr/adm/acct/nite/active
/usr/adm/acct/nite/daytacct
/usr/adm/acct/nite/lock
/usr/adm/acct/nite/lock1
/usr/adm/acct/nite/lastdate
/usr/adm/acct/nite/statefile
/usr/adm/acct/nite/ptacct*.mmd

SEE ALSO

acct(1M), acctcms(1M), acctcon(1M), acctmerg(1M), acctprc(1M), acctsh(1M), cron(1M), fwtmp(1M)
acctcom(1), mail(1) in the User's Reference Manual
BUGS

Normally it is not a good idea to restart `runacct` in the SETUP state. Run SETUP manually and restart via:

```
runacct mmd wtmpfix
```

If `runacct` failed in the PROCESS state, remove the last `ptacct` file because it will not be complete.
NAME
rwhod – system status server

SYNOPSIS
/usr/etc/rwhod [ -m [ ttl ] ]

DESCRIPTION
Rwhod is the server which maintains the database used by the rwho(1C)
and runtime(1C) programs. Its operation is predicated on the ability to
broadcast or multicast messages on a network. Rwhod is started at system
initialization if the configuration flag rwhod is set “on” with
chkconfig(1M). Site-dependent options and arguments to rwhod belong in
the file /etc/config/rwhod.options.

Rwhod operates as both a producer and consumer of status information. As
a producer of information it periodically queries the state of the system and
constructs status messages which are broadcast or multicast on a network.
As a consumer of information, it listens for other rwhod servers’ status mes-
sages, validating them, then recording them in a collection of files located in
the directory /usr/spool/rwho.

The server transmits and receives messages at the port indicated in the
“rwhod” service specification; see services(4). The messages sent and
received, are defined in <protocols/rwhod.h>:

```c
struct outmp {
    char out_line[8]; /* tty name */
    char out_name[8]; /* user id */
    long out_time; /* time on */
};

struct whod {
    char wd_vers; /* protocol version # */
    char wd_type; /* packet type: WHODTYPE_STATUS */
    char wd_pad[2];
    int wd_sendtime; /* time stamp by sender */
    int wd_recvtime; /* time stamp applied by receiver */
    char wd_hostname[32]; /* host's name */
    int wd_loadav[3]; /* load average as in uptime */
    int wd_boottime; /* time system booted */
    struct whoent {
        struct outmp we_utmp; /* active tty info */
        int we_idle; /* tty idle time */
    } wd_we[1024 / sizeof (struct whoent)];
};
```
All fields are converted to network byte order prior to transmission. The load averages represent smoothed CPU loads over the 5, 10, and 15 minute intervals prior to a server's transmission; they are multiplied by 100 for representation in an integer.

The host name included is that returned by the *gethostname(2)* system call, with any trailing domain name omitted. The array at the end of the message contains information about the users logged in to the sending machine. This information includes the contents of the *utmp(4)* entry for each non-idle terminal line and a value indicating the time in seconds since a character was last received on the terminal line.

Messages received by the *rwho* server are discarded unless they originated at an *rwho* server's port. In addition, if the host's name, as specified in the message, contains any unprintable ASCII characters, the message is discarded. Valid messages received by *rwhod* are placed in files named *whod.hostname* in the directory */usr/spool/rwho*. These files contain only the most recent message, in the format described above.

Status messages are generated approximately once every 3 minutes. *Rwhod* performs an *nlist(3)* on */unix* every 30 minutes to guard against the possibility that this file is not the system image currently operating.

*Rhwod* recognizes the following options:

- `-m` causes *rwhod* to use IP multicast (instead of broadcast or unicast) on all multicast-capable interfaces (excluding the loopback interface). The multicast reports are sent with a time-to-live of 1, to prevent forwarding beyond the directly-connected subnet(s).

- `-m ttl` causes *rwhod* to send IP multicast datagrams with a time-to-live of *ttl*, via the default multicast interface only rather than all interfaces. *ttl* must be between 0 and 32. Note that "-m 1" is different than "-m", in that "-m 1" specifies transmission on one interface only. This allows the information to be relayed between networks.

When "-m" is used without a *ttl* argument, the program accepts multicast *rwhod* reports from all multicast-capable interfaces. If a *ttl* argument is given, it accepts multicast reports from only one interface, the one on which reports are sent (which may be controlled via the host's routing table). Regardless of the "-m" option, the program accepts broadcast or unicast reports from all interfaces. Thus, this program will hear the reports of old, non-multicasting *rwhod*'s, but, if multicasting is used, those old *rwhod*'s won't hear the reports generated by this program.
SEE ALSO
   rwho(1C), ruptime(1C), rup(1C)

BUGS
   Status information should be sent only upon request rather than continuously. People often interpret the server dying or network communication failures as a machine going down.
NAME
sar: sa1, sa2, sadc – system activity report package

SYNOPSIS
/usr/lib/sa/sadc [t n] [ofile]
/usr/lib/sa/sa1 [t n]
/usr/lib/sa/sa2 [−ubdycwaqvmprtgA] [−s time] [−e time] [−i sec]

DESCRIPTION
System activity data can be accessed at the special request of a user (see sar(1)) and automatically on a routine basis as described here. The operating system contains a number of counters that are incremented as various system actions occur. These include counters for CPU utilization, buffer usage, disk and tape I/O activity, TTY device activity, switching and system-call activity, file-access, queue activity, inter-process communications, paging and Remote File Sharing.

Sadc and shell procedures, sa1 and sa2, are used to sample, save, and process this data.

Sadc, the data collector, samples system data n times every t seconds and writes in binary format to ofile or to standard output. If t and n are omitted, a special record is written. This facility is used at system boot time, when booting to a multiuser state, to mark the time at which the counters restart from zero. For example, the /etc/init.d/perf file writes the restart mark to the daily data by the command entry:

su sys −c "/usr/lib/sa/sadc /usr/adm/sa/sa−date +%d."

The shell script sa1, a variant of sadc, is used to collect and store data in binary file /usr/adm/sa/sadd where dd is the current day. The arguments t and n cause records to be written n times at an interval of t seconds, or once if omitted. The entries in /usr/spool/cron/crontabs/sys (see cron(1M)):

0 * * 0−6 /usr/lib/sa/sa1
20,40 8−17 * * 1−5 /usr/lib/sa/sa1

will produce records every 20 minutes during working hours and hourly otherwise.

The shell script sa2, a variant of sar(1), writes a daily report in file /usr/adm/sa/sardd. The options are explained in sar(1). The /usr/spool/cron/crontabs/sys entry:
will report important activities hourly during the working day.

The structure of the binary daily data file is:

```c
struct sa {
    struct sysinfo si;     /* see /usr/include/sys/sysinfo.h */
    struct minfo mi;       /* defined in sys/sysinfo.h */
    struct dinfo di;       /* RFS info defined in sys/sysinfo.h */
    int minserve, maxserve;  /* RFS server low and high water marks */
    int szinode;           /* current size of inode table */
    int szfile;            /* current size of file table */
    int szproc;            /* current size of proc table */
    int szlckf;            /* current size of file record header table */
    int szlckr;            /* current size of file record lock table */
    int mszinode;          /* size of inode table */
    int mszfile;           /* size of file table */
    int mszproc;           /* size of proc table */
    int mszlckf;           /* maximum size of file record header table */
    int mszlckr;           /* maximum size of file record lock table */
    long inodeovf;         /* cumulative overflows of inode table */
    long fileovf;          /* cumulative overflows of file table */
    long procovf;          /* cumulative overflows of proc table */
    time_t ts;             /* time stamp, seconds */
    long devio[NDEVS][4];  /* device unit information */
#define IO_OPS 0           /* cumulative I/O requests */
#define IO_BCNT 1          /* cumulative blocks transferred */
#define IO_ACT 2           /* cumulative drive busy time in ticks */
#define IO_RESP 3          /* cumulative I/O resp time in ticks */
};
```

FILES
/usr/adm/sa/sadd  daily data file
/usr/adm/sa/sardd  daily report file
/tmp/sa.adrfi  address file

SEE ALSO
cron(1M),
NAME
savecore — save a core dump of the operating system

SYNOPSIS
/etc/savecore [ -f ] [ -v ] dirname [ system ]

DESCRIPTION
savecore is meant to be called by /etc/rc2.d/S48savecore. Its function is to
save the core dump of the system (assuming one was made) and to write a
reboot message in the shutdown log. The S48savecore script specifies dir-
name as /usr/adm/crash by default, unless overridden by site-specific
command-line options in the file /etc/config/savecore.options.

savecore saves the core image in the file dirname/vmcore.n and its sibling,
the namelist, in dirname/unix.n The trailing ".n" in the pathnames is
replaced by a number which grows every time savecore is run in that direc-
tory.

Before savecore writes out a core image, it reads a number from the file
dirname/minfree. If the number of free kilobytes on the filesystem which
contains dirname is less than the number obtained from the minfree file, the
core dump is not saved. If the minfree file does not exist, savecore always
writes out the core file (assuming that a core dump was taken).

savecore also logs a reboot message using facility LOG_AUTH (see
syslog(3B)) If the system crashed as a result of a panic, savecore logs the
panic string too.

savecore assumes that /unix corresponds to the running system at the time
of the crash. If the core dump was from a system other than /unix, the name
of that system must be supplied as system.

The following options apply to savecore:

- f Ordinarily, savecore checks a magic number on the dump device
  (usually /dev/swap ) to determine if a core dump was made. This
  flag will force savecore to attempt to save the core image regard-
  less of the state of this magic number. This may be necessary
  since savecore will always clear the magic number after reading it.
  If a previous attempt to save the image failed in some manner, it
  will still be possible to restart the save with this option.

- v Give more verbose output.

DIAGNOSTICS
"warning: /unix may not have created core file" is printed if savecore
believes that the system core file does not correspond with the /unix operat-
ing system binary.
"savecore: /unix is not the running system" is printed for the obvious reason. If the system which crashed was /unix use mv(1) to change its name before running savecore. Use mv(1) or ln(1) to rename or produce a link to the name of the file of the currently running operating system binary such that savecore can find name list information about the current state of the running system from the file /unix.

FILES

/unix
/usr/adm/crash
/etc/config/savecore.options

current UNIX
default location
site-specific command-line options

SEE ALSO

nlist(3X)
NAME
sendmail, newaliases, mailq – send mail over the internet

SYNOPSIS
/usr/lib/sendmail [ flags ] [ address ... ]
newaliases
mailq

DESCRIPTION
sendmail sends a message to one or more people, routing the message over whatever networks are necessary. sendmail does internetwork forwarding as necessary to deliver the message to the correct place.

sendmail is not intended as a user interface routine; other programs provide user-friendly front ends; sendmail is used only to deliver pre-formatted messages.

With no flags, sendmail reads its standard input up to a control-D or a line with a single dot and sends a copy of the letter found there to all of the addresses listed. It determines the network to use based on the syntax and contents of the addresses.

Local addresses are looked up in a file and aliased appropriately. Aliasing can be prevented by preceding the address with a backslash. Normally the sender is not included in any alias expansions, e.g., if 'john' sends to 'group', and 'group' includes 'john' in the expansion, then the letter will not be delivered to 'john'.

Flags are:

- ba Go into ARPANET mode. All input lines must end with a CR-LF, and all messages will be generated with a CR-LF at the end. Also, the "From:" and "Sender:" fields are examined for the name of the sender.

- bd Run as a daemon. This requires Berkeley IPC.

- bi Initialize the alias database.

- bm Deliver mail in the usual way (default).

- bp Print a listing of the queue.

- bs Use the SMTP protocol as described in RFC821. This flag implies all the operations of the - ba flag that are compatible with SMTP.

- bt Run in address test mode. This mode reads addresses and shows the steps in parsing; it is used for debugging configuration tables.
-bv Verify names only – do not try to collect or deliver a message. Verify mode is normally used for validating users or mailing lists.

-bz Create the configuration freeze file.

-Cfile
Use alternate configuration file.

-dX Set debugging value to X.

-Ffullname
Set the full name of the sender.

-fname
Sets the name of the "from" person (i.e., the sender of the mail). -f can only be used by the special users root, daemon, and network, or if the person you are trying to become is the same as the person you are.

-hN Set the hop count to N. The hop count is incremented every time the mail is processed. When it reaches a limit, the mail is returned with an error message, the victim of an aliasing loop.

-n Don’t do aliasing.

-ox value
Set option x to the specified value. Options are described below.

-q[time]
Processed saved messages in the queue at given intervals. If time is omitted, process the queue once. Time is given as a tagged number, with 's' being seconds, 'm' being minutes, 'h' being hours, 'd' being days, and 'w' being weeks. For example, "-q1h30m" or "-q90m" would both set the timeout to one hour thirty minutes.

-rname
An alternate and obsolete form of the -f flag.

-t Read message for recipients. To:, Cc:, and Bcc: lines will be scanned for people to send to. The Bcc: line will be deleted before transmission. Any addresses in the argument list will be suppressed.

-v Go into verbose mode. Alias expansions will be announced, etc.

There are also a number of processing options that may be set. Normally these will only be used by a system administrator. Options may be set either on the command line using the -o flag or in the configuration file.

These are described in detail in the Installation and Operation Guide. The options are:
Afile  Use alternate alias file.

C  On mailers that are considered "expensive" to connect to, don't initiate immediate connection. This requires queueing.

dx  Set the delivery mode to x. Delivery modes are 'i' for interactive (synchronous) delivery, 'b' for background (asynchronous) delivery, and 'q' for queue only – i.e., actual delivery is done the next time the queue is run.

D  Try to automatically rebuild the alias database if necessary.

ex  Set error processing to mode x. Valid modes are 'm' to mail back the error message, 'w' to "write" back the error message (or mail it back if the sender is not logged in), 'p' to print the errors on the terminal (default), 'q' to throw away error messages (only exit status is returned), and 'e' to do special processing for the BerkNet. If the text of the message is not mailed back by modes 'm' or 'w' and if the sender is local to this machine, a copy of the message is appended to the file "dead.letter" in the sender's home directory.

Fmode
   The mode to use when creating temporary files.

f  Save UNIX-style From lines at the front of messages.

gN  The default group id to use when calling mailers.

Hfile  The SMTP help file.

i  Do not take dots on a line by themselves as a message terminator.

Ln  The log level.

m  Send to "me" (the sender) also if I am in an alias expansion.

o  If set, this message may have old style headers. If not set, this message is guaranteed to have new style headers (i.e., commas instead of spaces between addresses). If set, an adaptive algorithm is used that will correctly determine the header format in most cases.

Quedir
   Select the directory in which to queue messages.

rtimeout
   The timeout on reads; if none is set, sendmail will wait forever for a mailer.

Sfile  Save statistics in the named file.
s Always instantiate the queue file, even under circumstances where it is not strictly necessary.

Ttime
Set the timeout on messages in the queue to the specified time. After sitting in the queue for this amount of time, they will be returned to the sender. The default is three days.

tsz,dtz
Set the name of the time zone.

uN Set the default user id for mailers.

If the first character of the user name is a vertical bar, the rest of the user name is used as the name of a program to pipe the mail to. It may be necessary to quote the name of the user to keep sendmail from suppressing the blanks from between arguments.

Sendmail returns an exit status describing what it did. The codes are defined in <sysexits.h>

- EX_OK Successful completion on all addresses.
- EX_NOUSER User name not recognized.
- EX_UNAVAILABLE Catchall meaning necessary resources were not available.
- EX_SYNTAX Syntax error in address.
- EX_SOFTWARE Internal software error, including bad arguments.
- EX_OSERR Temporary operating system error, such as "cannot fork".
- EX_NOHOST Host name not recognized.
- EX_TEMPFAIL Message could not be sent immediately, but was queued.

If invoked as newaliases, sendmail will rebuild the alias database. If invoked as mailq, sendmail will print the contents of the mail queue.

FILES
Except for /usr/lib/sendmail.cf, these pathnames are all specified in /usr/lib/sendmail.cf. Thus, these values are only approximations.

- /usr/lib/aliases raw data for alias names
- /usr/lib/aliases.pag data base of alias names
- /usr/lib/aliases.dir configuration file
- /usr/lib/sendmail.cf frozen configuration
- /usr/lib/sendmail.fc help file
- /usr/lib/sendmail.hf collected statistics
- /usr/lib/sendmail.st to deliver uucp mail
- /usr/bin/uux

April 1990 - 4 - Version 5.0
SEE ALSO

mail_att(1), mail_bsd(1), rmail(1M), newaliases(1M), syslog(3), aliases(4),
rc2(1M);
Internet Request For Comments RFC819, RFC821, RFC822;
Sendmail – An Internetwork Mail Router;
Sendmail Installation and Operation Guide.

BUGS

Sendmail converts blanks in addresses to dots. This is incorrect according
to the old ARPANET mail protocol RFC733 (NIC 41952), but is consistent
with the new protocols (RFC822).
NAME

setmnt – establish mount table

SYNOPSIS

/etc/setmnt

DESCRIPTION

setmnt creates the /etc/mnttab table which is needed for both the
mount(1M) and umount commands. setmnt reads standard input and
creates a mnttab entry for each line. Input lines have the format:

    filesystem node

where filesystem is the name of the file system’s special file (e.g.,
/dev/dsk/c?d?s?) and node is the root name of that file system. Thus
filesystem and node become the first two strings in the mount table entry.

FILES

/etc/mtab

SEE ALSO

devnm(1M), mount(1M).

BUGS

Problems may occur if filesystem or node are longer than 32 characters.
setmnt silently enforces an upper limit on the maximum number of mnttab
entries.

NAME
setsym – set up a debug kernel for symbolic debugging

SYNOPSIS
setsym [-v] [filename]

DESCRIPTION
setsym reads in the symbol table of the file 'filename' and writes it back into
an internal symbol table that is part of 'filename' .data section. The input
file is intended to be a debug kernel. This internal symbol table is used by
symmon for symbolic debugging of the kernel.

By default, filename is 'unix'. Additional error messages can be seen with
the -v flag. The size of the internal symbol table is hardwired into setsym
and the kernel. Currently, the table size is fixed at 4800 symbol entries.

SEE ALSO
lboot(1m).
SHUTDOWN(1M)  Silicon Graphics  SHUTDOWN(1M)

NAME
shut down – shut down system, change system state

SYNOPSIS
  cd /; /etc/shutdown [ -y ] [ -ggrace_period [ -iinit_state ]

DESCRIPTION
This command is executed by the super-user to change the state of the machine. By default, it brings the system down into the PROM monitor.

The command sends a warning message and a final message before it starts actual shutdown activities. By default, the command asks for confirmation before it starts shutting down daemons and killing processes. The options are used as follows:

  -y  pre-answers the confirmation question so the command can be run without user intervention. A default of 60 seconds is allowed between the warning message and the final message. Another 60 seconds is allowed between the final message and the confirmation.

  -ggrace_period
allows the super-user to change the number of seconds from the 60-second default.

  -iinit_state
specifies the state that init(1M) is to be put in following the warnings, if any. By default, system state "0" is used.

The definitions of the init_state values that shutdown will accept are:

state 0
  Stop the UNIX system and go to the firmware monitor. At this point it is safe to power down the machine. The /etc/rc0 procedure is called upon entering this init_state to perform various system cleanup functions.

state 1, s, S
  Bring the machine to the state traditionally called single-user, which provides an acquiescent system and a superuser shell running on the system console. Upon entering this state, the system kills all user processes and unmounts all file systems except the root. The /etc/rc0 procedure is called to do these and various other system cleanup functions.

state 6
  Stop the UNIX system and reboot to the state defined by the initdefaul t entry in /etc/inittab.
SEE ALSO

init(1M), rc0(1M), rc2(1M).
NAME
single – switch the system to single-user mode

SYNOPSIS
/etc/single

DESCRIPTION
Single switches the system to single-user mode and turns the gettys off. Single does not kill background processes such as cron, or update. These must be killed with kill or killall. Single is a shell script that invokes /etc/telinit.

SEE ALSO
multi(1M), init(1M), getty(1M), kill(1), killall(1M), inittab(4).
NAME

su – become super-user or another user

SYNOPSIS

su [ - ] [ name [ arg ... ] ]

DESCRIPTION

su allows one to become another user without logging off. The default user
name is root (i.e., super-user).

To use su, the appropriate password must be supplied (except as described
below). If the password is correct, su will execute a new shell with the real
and effective user ID set to that of the specified user. The new shell will be
the optional program named in the shell field of the specified user’s pass-
word file entry (see passwd(4)), or /bin/sh if none is specified (see sh(1)).
To restore normal user ID privileges, type an EOF (cntrl-d) to the new shell.

Su prompts for a password if the specified user’s account has one. How-
ever, su will not prompt you if your user name is root or your name is listed
in the specified user’s .rhosts file as:

    localhost your_name

Any additional arguments given on the command line are passed to the pro-
gram invoked as the shell. When using programs like sh(1), an arg of the
form –c string executes string via the shell and an arg of –r will give the
user a restricted shell.

The following statements are true only if the optional program named in the
shell field of the specified user’s password file entry is like sh(1). If the first
argument to su is a –, the environment will be changed to what would be
expected if the user actually logged in as the specified user. This is done by
invoking the program used as the shell with an arg0 value whose first char-
acter is –,

thus causing first the system’s profile (/etc/profile) and then the specified
user’s profile (.profile in the new HOME directory) to be executed.

Otherwise, the environment is passed along with the possible exception of
$PATH, which is set to

/usr/sbin:/usr/bsd:/bin:/etc:/usr/bin:/usr/etc:/usr/bin/X11

for root. Note that if the optional program used as the shell is /bin/sh, the
user’s .profile can check arg0 for –sh or –su to determine if it was invoked
by login(1) or su(1), respectively. If the user’s program is other than
/bin/sh, then .profile is invoked with an arg0 of –program by both login(1)
and su(1).
All attempts to become another user using `su` are logged in the log file `/usr/adm/sulog`.

**EXAMPLES**

To become user `bin` while retaining your previously exported environment, execute:

```
su bin
```

To become user `bin` but change the environment to what would be expected if `bin` had originally logged in, execute:

```
su - bin
```

To execute *command* with the temporary environment and permissions of user `bin`, type:

```
su - bin -c "command args"
```

**FILES**

- `/etc/passwd` system’s password file
- `/etc/profile` system’s initialization script for `/bin/sh` users
- `/etc/cshrc` system’s initialization script for `/bin/csh` users
- `$HOME/.profile` `/bin/sh` user’s initialization script
- `$HOME/.cshrc` `/bin/csh` user’s initialization script
- `$HOME/.rhosts` user’s list of trusted users
- `/usr/adm/sulog` log file

**SEE ALSO**

`env(1)`, `login(1)` in the *User’s Reference Manual*.
`passwd(4)`, `cshrc(4)`, `profile(4)`, `rhosts(4)`, `environ(5)` in the *Programmer’s Reference Manual*.
NAME

swap – swap administrative interface

SYNOPSIS

/etc/swap -a swapdev swaplow swaplen
/etc/swap -d swapdev swaplow
/etc/swap -l

DESCRIPTION

swap provides a method of adding, deleting, and monitoring the system swap areas used by the memory manager. The following options are recognized:

-a Add the specified swap area. swapdev is the name of the block special device. swaplow is the offset in 512-byte blocks into the device where the swap area should begin. swaplen is the length of the swap area in 512-byte blocks. This option can only be used by the super-user. Swap areas are normally added by the system start up routines in /etc/rc2.d when going into multi-user mode.

-d Delete the specified swap area. swapdev is the name of block special device, e.g., /dev/dsk/dks0d2s1. swaplow is the offset in 512-byte blocks into the device where the swap area should begin. Using this option marks the swap area as "INDEL" (in process of being deleted). The system will not allocate any new blocks from the area, and will try to free swap blocks from it. The area will remain in use until all blocks from it are freed. This option can only be used by the super-user.

-l List the status of all the swap areas. The output has four columns:

DEV  The swapdev special file for the swap area if one can be found in the /dev/dsk or /dev directories, and its major/minor device number in decimal.

LOW  The swaplow value for the area in 512-byte blocks.

LEN  The swaplen value for the area in 512-byte blocks.

FREE The number of free 512-byte blocks in the area. If the swap area is being deleted, this column will be marked INDEL.

WARNINGS

No check is done to see if a swap area being added overlaps with an existing swap area or file system.
NAME
  sync – update the super block

SYNOPSIS
  sync

DESCRIPTION
  sync executes the sync system primitive. If the system is to be stopped,
sync must be called to insure file system integrity. It will flush all previ-
ously unwritten system buffers out to disk, thus assuring that all file
modifications up to that point will be saved. See sync(2) for details.

NOTES
  If you have done a write to a file on a remote machine using NFS, you can-
not use sync to force buffers to be written out to disk on the remote
machine. sync will only write local buffers to local disks.

  Note that the sync command may complete and the user may get another
shell prompt before the flushing of memory buffers to disk is complete.
sync merely schedules the buffers to be written before exiting.

SEE ALSO
NAME
syslogd – log systems messages

SYNOPSIS
/usr/etc/syslogd [ -f configfile ] [ -m markinterval ] [ -p logpipe ] [ -d ]

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

Syslogd reads its configuration when it starts up and whenever it receives a hangup 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(s) are separated from the selector by one or more tabs.

Selectors are semicolon separated lists of priority specifiers. 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. Symbolic names may be used. An asterisk selects all facilities, while "debug" selects all levels. All messages of the specified level or higher (greater severity) are selected. More than one facility may be selected using commas to separate them. For example:

*.emerg;mail,daemon.crit

Selects all facilities at the emerg level and the mail and daemon facilities at the crit level.

Known facilities and levels recognized by syslogd are those listed in syslog(3) without the leading "LOG_". The additional facility "mark" logs messages at priority LOG_INFO every 20 minutes (this interval may be changed with the -m flag). The "mark" facility is not enabled by a facility field containing an asterisk. The level "none" may be used to disable a particular facility. For example,

*.debug;mail.none

Sends all messages except mail messages to the selected file.

The second part of each line describes where the message is to be logged if this line is selected. There are five forms:
- A filename (beginning with a leading slash). The file will be opened in append mode.

- A hostname preceded by an at sign ("@"). Selected messages are forwarded to the syslogd on the named 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.

- A l, followed immediately by a program name, which is taken to be all chars after the l, up to the next tab; at least one action must follow the tab. The filter is expected to read stdin, and write the filtered response to stdout. If the filter exits with a non-zero value, the original message is logged, as well as a message that the filter failed. The filter has a limited time (currently 8 seconds) to process the message. If the filter exits with status 0 without writing any data, no message is logged. The data to be read by the filter is not terminated with a newline, nor should the data written have a newline appended.

Blank lines and lines beginning with ‘#’ are ignored.

For example, the configuration file:

```
kern.debug  |/usr/adm/klogpp  /usr/adm/SYSLOG
kern.debug  |/usr/adm/klogpp  /dev/console
user,mail,daemon,auth,syslog,lpr.debug  /usr/adm/SYSLOG
kern.err    @ginger
*.emerg      *
*.alert      eric,beth
*.alert;auth.warning  ralph
```

filters all kernel messages through /usr/adm/klogpp and writes them to the system console and into /usr/adm/SYSLOG, logs debug (or higher) level messages into the file /usr/adm/SYSLOG; kernel messages of error severity or higher are forwarded to ginger. All users will be informed of any emergency messages, the users “eric” and “beth” will be informed of any alert messages, and the user “ralph” will be informed of any alert message, or any warning message (or higher) from the authorization system.

Syslogd is started at system initialization from /etc/init.d/syslogd. Optional site-specific flags belong in /etc/config/syslogd.options. The flags are:

- `f` Specify an alternate configuration file.
- `m` Select the number of minutes between mark messages.
-d Turn on debugging. syslogd runs in the foreground and writes debugging info to stdout.

-p Use the given name for the named pipe, instead of /dev/log.

Syslogd rereads its configuration file when it receives a hangup signal, SIGHUP. To bring syslogd down, it should be sent a terminate signal (e.g., killall –TERM syslogd).

FILES

/etc/syslog.conf Default configuration file
/dev/log Named pipe read by syslogd (created at startup and removed at termination)
/dev/klog The kernel log device
/usr/adm/klogpp Filter for kernel messages
/etc/config/syslogd.options Command-line flags used at system startup

SEE ALSO

syslog(3)
NAME

tabled – tablet reader daemon for Bitpad I compatible tablet/digitizers

SYNOPSIS

/etc/gl/tabled /dev/ttyd? [resolution]

DESCRIPTION

This is the tablet reader for the Hitachi tablet. The Hitachi tablet is 11" x 11" and has 200 points per inch. Without specifying a resolution argument, tablet values generated will span the range of 0 to 2200 in X and Y. If the cursor is attached to the tablet, the lower left-hand corner of the tablet maps into the screen area, since tablet values exceed XMAXSCREEN and YMAXSCREEN.

By specifying a resolution value--most commonly this would be 2200--tablet values will be between 0 and XMAXSCREEN in X and between 0 and XMAXSCREEN in Y. If the cursor is attached to the tablet, the whole bottom area of the tablet maps into the screen area. This will map the entire width of the tablet onto the screen. Other tablets resolutions may vary and thus the correct size in X must be used to accurately map each tablet to full XMAXSCREEN width. Since the 4D graphics console is longer in X than in Y, the topmost part of the tablet will still be "out of range" with respect to screen coordinates if a resolution of 2200 is used.
NAME
talkd – remote user communication server

SYNOPSIS
/usr/etc/talkd

DESCRIPTION
Talkd is the server that notifies a user that somebody else wants to initiate a conversation. It acts a repository of invitations, responding to requests by clients wishing to rendezvous to hold a conversation. In normal operation, a client, the caller, initiates a rendezvous by sending a CTL_MSG to the server of type LOOK_UP (see <protocols/talkd.h>). This causes the server to search its invitation tables to check if an invitation currently exists for the caller (to speak to the callee specified in the message). If the lookup fails, the caller then sends an ANNOUNCE message causing the server to broadcast an announcement on the callee’s login ports requesting contact. When the callee responds, the local server uses the recorded invitation to respond with the appropriate rendezvous address and the caller and callee client programs establish a stream connection through which the conversation takes place.

SEE ALSO
talk(1), write(1)
NAME
telnetd – Internet TELNET protocol server

SYNOPSIS
/usr/etc/telnetd

DESCRIPTION
Telnetd is a server which supports the Internet standard TELNET virtual
terminal protocol. Telnetd is invoked by the Internet super-server (see
inetd(1M)), normally for requests to connect to the TELNET port as indi-
cated by the /etc/services file (see services(4)).

Telnetd operates by allocating a pseudo-terminal device (see pty(7)) for a
client, then creating a login process which has the slave side of the pseudo-
terminal as stdin, stdout, and stderr. Telnetd manipulates the master side
of the pseudo-terminal, implementing the TELNET protocol and passing
characters between the remote client and the login process.

When a TELNET session is started up, telnetd sends TELNET options to
the client side indicating a willingness to do remote echo of characters, to
suppress go ahead, to do remote flow control, and to receive terminal type
information, terminal speed information, and window size information from
the remote client. If the remote client is willing, the remote terminal type is
propagated in the environment of the created login process.

Telnetd is willing to do: echo, binary, suppress go ahead, and timing
mark. Telnetd is willing to have the remote client do: binary, terminal
type, terminal speed, window size, toggle flow control, and suppress go
ahead.

SEE ALSO
telnet(1C)

BUGS
Some TELNET commands are only partially implemented.

Because of bugs in the original 4.2 BSD telnet(1C), telnetd performs some
dubious protocol exchanges to try to discover if the remote client is, in fact,
a 4.2 BSD telnet(1C).

Binary mode has no common interpretation except between similar operat-
ing systems (Unix in this case).

The terminal type name received from the remote client is converted to
lower case.

Telnetd never sends TELNET go ahead commands.
NAME
tftp – Internet Trivial File Transfer Protocol server

SYNOPSIS
/usr/etc/tftp [-h homedir] [-l] [-n] [-s] [directory... ]

DESCRIPTION
Tftp is a server which supports the Internet Trivial File Transfer Protocol.
The TFTP server operates at the port indicated in the “tftp” service
description; see services(4). The server is normally started by inetd(1M).

The use of tftp does not require an account or password on the remote system. Due to the lack of authentication information, tftp will allow only
publicly readable files to be accessed. Files containing the string ‘‘..’’ are
not allowed. Files may be written only if they already exist, and are publicly writable. Note that 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 tftp service. The server should be configured in
/usr/etc/inetd.conf to run as the user ID with the lowest possible privilege.

Relative filenames are looked up in a home directory, /usr/etc/boot by
default. The –h option changes the home directory to homedir, provided it
is an absolute pathname. The –l option logs all requests using syslog(3).
The –n option suppresses negative acknowledgement of requests for nonex-
istent or inaccessible relative filenames. Use –n when operating on a net-
work with Sun diskless clients that broadcast TFTP requests for bootfiles
named by relative pathnames, to avoid storms of negative acknowledge-
ments.

Normally, tftp allows unrestricted access to publicly-readable files in all
directories. There are two ways to enhance file security by restricting
access to a smaller set of directories. With the –s option, tftp will reject
requests to read or write an absolute pathname that does not begin with the
home directory prefix. Another method is to restrict access to files in a lim-
ited number of “approved” directories by specifying the directory names as arguments to tftp after the other options. For an absolute pathname
request, tftp allows the request if its name begins with one of these direc-
tories or the home directory. For a relative pathname request, the home
directory and the directory list are searched in order. Up to 10 directories
can be listed if no other command-line options are specified. (Inetd limits
the total number of command-line arguments to 10).

SEE ALSO
tftp(1C), inetd(1M)
NAME
tic — terminfo compiler

SYNOPSIS
tic [-v[n]] [-c] file

DESCRIPTION
tic translates a terminfo(4) 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(3X).

-\*n (verbose) output to standard error trace information showing tic’s progress. 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 only check file for errors. Errors in use= links are not detected.

file contains one or more terminfo(4) terminal descriptions in source format (see terminfo(4)). 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 environment variable TERMINFO 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 explicitly are defined in the current entry.

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

FILES
/usr/lib/terminfo/* compiled terminal description data base

SEE ALSO

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

Terminal names exceeding 14 characters will be truncated to 14 characters and a warning message will be printed.

April 1990

Version 5.0
When the \(-c\) option is used, duplicate terminal names will not be diagnosed; however, when \(-c\) is not used, they will be.

**BUGS**

To allow existing executables from the previous release of the UNIX 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 \texttt{use= entry}. Such names would not be used for real terminal names.)

For example:

\[
\begin{align*}
4415+n1, & \text{ kf1@, kf2@, ...} \\
4415+\text{base, & kf1=EOc, kf2=EOd, ...} \\
4415-\text{nl} & \text{4415 terminal without keys,} \\
\text{use=4415+nl, use=4415+base,} \\
\end{align*}
\]

The above example works as expected; the definitions for the keys do not show up in the 4415-\texttt{nl} entry. However, if the entry 4415+\texttt{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-\texttt{nl}.

**DIAGNOSTICS**

Most diagnostic messages produced by \texttt{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.

\texttt{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 \texttt{seek(2)} not \texttt{NAMES}
   Somehow the file being compiled changed during the compilation.

Not enough memory for \texttt{use_list} element
   or

Out of memory
   Not enough free memory was available (\texttt{malloc(3)} 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
or
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 to 14 characters due to UNIX 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.

Newline in middle of terminal name
   The trailing comma was probably left off of 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 invalid option was entered.

Too many file names. Usage is:
   self-explanatory

"...
non-existent 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 non-existent terminals. The following is a list of the entries
involved:

   A terminfo(4) entry with a use=name capability either referenced a
   non-existent terminal called name or name somehow referred back
to the given entry.
NAME

timed – time server daemon

SYNOPSIS

/usr/etc/timed [ -tM ] [ -G netgroup ] [ -F host1 host2 ... ]
[ -n network ] [ -i network ] [ -P param-file ]

DESCRIPTION

Timed is the time server daemon and is normally invoked at boot time from the /etc/inittab/network file. It synchronizes the host’s time with the time of other machines in a local area network running timed. These time servers will slow down the clocks of some machines and speed up the clocks of others to bring them to the average network time. The average network time is computed from measurements of clock differences using the ICMP timestamp request message.

The service provided by timed is based on a master-slave scheme. When timed is started on a machine, it asks the master for the network time and sets the host’s clock to that time. After that, it accepts synchronization messages periodically sent by the master and calls adjtime(2) to perform the needed corrections on the host’s clock. The master adjusts its own clock by averaging the clocks of all trusted machines (see below) with its own clock.

Timed communicates with date(1) in order to set the date globally, and with timedc(1M), a timed control program. If the machine running the master crashes, then the slaves will elect a new master from among slaves running with the -M flag. A timed running without the -M, -F, and -G flags will remain a slave.

In addition, timed logs accumulated corrections in the system log (see syslogd(1M)) to ease adjusting the local clock. The clock can be adjusted by changing the timetrtrim parameter in the master.d/kernel file or with the syssgi(2) system call. This parameter is used by the operating system to compensate for variations among machines. It can be used to improve the accuracy of the local clock. -P param-file specifies a file in which to save a computed value of the timetrtrim parameter. The active value in the operating system is set to the value found in the file when the daemon started. A good choice for the file name is /usr/tmp/timetrtrim.

The -F flag means only the local machine and the machines host1, host2, etc., are trusted to have good hardware clocks and to be securely administered. Any attempts to change the clocks by other, untrusted machines are ignored, except to log them in the system log. The clocks of untrusted machines are not averaged by the master. A master which has been told it is trustworthy will tell untrustworthy machines which try to be elected master to be slaves. The clock of a daemon started with -F localhost "free runs."
The `-G` flag is used to specify a netgroup (see `netgroup(4)`) of trusted machines. This flag allows central administration of the list of trusted machines. Making gateways trusted ties the clocks in an internet together, because after a network partition is healed, the trusted gateways will suppress the upstart local master elected during the partition.

`Timed` requests synchronization service from the first master server located. If permitted by the `-M` flag, it will provide synchronization service on any attached networks on which no current master server was detected. Such a server propagates the time computed by the top-level master. The `-n` flag, followed by the name of a network which the host is connected to (see `networks(4)`), overrides the default choice of the network addresses made by the program. Each time the `-n` flag appears, that network name is added to a list of valid networks. All other networks are ignored. The `-i` flag, followed by the name of a network to which the host is connected (see `networks(4)`), overrides the default choice of the network addresses made by the program. Each time the `-i` flag appears, that network name is added to a list of networks to ignore. All other networks are used by the time daemon. The `-n` and `-i` flags are meaningless if used together.

The `-t` flag causes `timed` to trace the messages it receives in the file `/usr/adm/timed.log`. Tracing can be turned on or off by the program `timedc(1M)`. `Timed` normally checks for a master time server on each network to which it is connected, except as modified by the options described below.

The `timeslave(1M)` command can be used to inexpensively synchronize the clock on a machine to the clock on a remote machine. It does not require any daemons or special programs on the remote machine.

FILES

/`usr/adm/timed.log` tracing file for timed
/`usr/adm/timed.masterlog` log file for master timed
/`usr/adm/SYSLOG` system log
/etc/init.d/network start-up script
/etc/config/timed.options optional flags, by default
"-G timelords -P /usr/tmp/.timetrim"

SEE ALSO

chkconfig(1M), date(1), adjtime(2), gettimeofday(2), icmp(4P),
timedc(1M), timeslave(1M)
NAME
timedc – timed control program

SYNOPSIS
/usr/etc/timedc [ command [ argument ... ] ]

DESCRIPTION
Timedc is used to control the operation of the timed(1M) program. It may be used to:
• measure the differences between machines’ clocks,
• find the location where the master time server is running,
• enable or disable tracing of messages received by timed, and
• perform various debugging actions.

Without any arguments, timedc will prompt for commands from the standard input. If arguments are supplied, timedc interprets the first argument as a command and the remaining arguments as parameters to the command. The standard input may be redirected causing timedc to read commands from a file. Commands may be abbreviated; recognized commands are:
?
[ command ... ]
help [ command ... ]
Print a short description of each command specified in the argument list, or, if no arguments are given, a list of the recognized commands.
clockdiff host ...
Compute the differences between the clock of the host machine and the clocks of the machines given as arguments.
trace [ on | off ]
Enable or disable the tracing of incoming messages to timed in the file /usr/adm/timed.log.
election host
Asks the timed on the target host to reset its "election" timers and to ensure that a time master has been elected.
quit
Exit from timedc.

FILES
/usr/adm/timed.log tracing file for timed
/usr/adm/timed.masterlog log file for master timed

April 1990 - 1 - Version 5.0
SEE ALSO
  date(1), timed(1M), timeslave(1M), adjtime(2), icmp(7P)

DIAGNOSTICS
  ?Ambiguous command  abbreviation matches more than one command
  ?Invalid command    no match found
  ?Privileged command command can be executed by root only
NAME
timeslave - 'slave' local clock to a better one

SYNOPSIS

    timeslave [ options ] -H netmaster
    timeslave [ options ] -C tty

where options can be any of:

    [ -m ] [ -d ] [ -B ] [ -w icmp-wait ] [ -W time-wait ]
    [ -t min-trials ] [ -T max-trials ] [ -r rate ]
    [ -p port ] [ -D max-drift ] [ -i init-drift ]
    [ -P param-file ]

DESCRIPTION

Timeslave matches the clock in the local machine to a better clock. It does
this by speeding up or slowing down the local clock, or if the local clock is
particularly wrong, by changing the date. When the date is changed,
because the difference is too great to correct smoothly, timeslave logs the
event in the system log.

Timeslave understands several options:

-H netmaster
    specifies the hostname or Internet address of another machine that
    has a better clock or that has, in turn, slaved its clock to a better
    one. Either a host must be specified with -H or a device must be
    specified with -C, but not both.

-C tty
    specifies the filename of a tty port, such as /dev/ttyd2, with a Preci-
    sion Time Standards WWV receiver.

-P param-file
    specifies a file in which to save a computed value of the timetrim
    parameter. The active value in the operating system is set to the
    value found in the file when the deamon started. A good choice
    for the file name is /usr/tmp/.timetrim.

It is usually not necessary to use the following options to change the default
values compiled in timeslave:

-m
    causes timeslave to measure the difference between the local and
    remote clock (or WWV receiver), but to not change the local
    clock.

-d
    increases the 'debugging level,' causing more messages to appear
    in the system log. Specifying this option several times increases
    the amount of debugging information. The signals SIGUSR1 and
    SIGUSR2 can be used to increase and decrease, respectively, the
    current amount of debugging information.
-B disables the normal 'backgrounding' of the deamon.

-w icmp-wait
is the number of seconds before an ICMP timestamp packet is assumed to have been lost in the network. -W time-wait the number of seconds before the remote machine is assumed to be not answering time service requests. The first time the remote machine fails to answer a time service request is logged in the system log. Additional failures are also logged if debugging is requested.

-t min-trials
specifies the minimum number of ICMP packets to use to determine the time difference.

-T max-trials
specifies the maximum number of ICMP packets to use to determine the time difference. If the remote machine does not respond to this many requests, it is assumed to be temporarily dead.

-r rate specifies the approximate number of seconds between measurements. The time between one measurement and the next is changed slightly each time to avoid systematic errors caused by other activity in the machines or the network in general. For example, cron starts a request in the first few seconds of a minute.

-p port is the port number on which the other machine provides Internet time service, as describe in RFC-868.

-D max-drift
specifies the maximum rate at which the local and remote clocks can be expected to drift, in nanoseconds per second. This is used to discard bad measurements. Its default value is 1% or 10,000,000 nanoseconds per second.

-i init-drift
specifies an initial value of the drift of the local clock in nanoseconds per second. This value can be used to make timeslave better behaved when it is first started.

Timeslave consumes less network bandwidth, fewer CPU cycles and less memory on both the local and remote machines than timed(1M). Timed is appropriate for a group of peers which average their equivalently accurate clocks to find a better estimate of the correct time. Because the master of the network time is elected by the timed participants, more correct time can be maintained despite the failure of individual systems.
A small, homogeneous group of machines should use timed, rather than timeslave. In a large network, one should use a small group of trusted, well-administered machines running timed with the -F or -G option to exclude other machines. The excluded machines should run timed with only the -M flag. One of the trusted machines should, if possible, be slaved with timeslave to a WWV receiver, or some remote machine with a very accurate clock. This machine should be lightly loaded, and should use the -F option with timed to make itself the most trusted time keeper in the local network. This scheme, a “king” with a circle of trusted substitutes surrounded by a larger number of machines should be replicated on each subnetwork in an installation, building a hierarchy of domains. The local king can use timeslave to synchronize its clock to very remote machines over low-performance, wide-area networks such as the DARPA Internet.

A subnetwork consisting of two or more sub-subnetworks, such as a group of ethernets, can be synchronized in a similar fashion. The king of the subnetwork would be synchronized with timeslave to a remote network or a WWV receiver (see below). The circle of trusted substitutes would be the gateways to the other sub-subnetworks.

Timeslave operates in one of two modes. In the first, it matches the clock in the local machine to the clock in some other machine. It sends UDP datagrams to the ’time’ service on the other machine to determine the current day. The time service on machines using 4.3BSD-style networking is provided by the inetd(1M) deamon. This allows timeslave to determine the correct day. Timeslave then uses ICMP timestamp packets to measure difference between the local and remote clocks in milliseconds. It assumes that the round trip delay for packets to and from the remote machine is symmetric. This is usually a valid assumption. However, large network data transfers can make transmission delays tens of thousands of times larger or smaller than reception delays. Therefore, timeslave elaborately filters and averages its measurements. In addition, it lowers its scheduling priority to minimize the effects of other programs on the local machine.

The second mode in which timeslave can operate is with a Precision Time Standards WWV receiver. In this case, it still must average its measurements to compensate for variable delays due to the operating system. When using a clock, it is best to minimize any latencies in the serial line connected to the clock, by adjusting the relavent parameters for the driver. See cfsio(1M) and sduart(1M).

Timeslave logs accumulated corrections in the system log (see syslogd(1M) to ease adjusting the local clock. The clock can be adjust by changing the timetrim parameter in the master.d/kernel file. This parameter is used by the operating system to compensate for variations among machines. It can be used to improve the accuracy of the local clock.
Timeslave immediately replaces the value of the timetrim parameter in
master.dl with the contents of the file specified with the -P option, and after
measuring for several hours chooses a better value. Besides logging the
value, timeslave periodically writes the value to the file specified with -P.

FILES
/usr/adm/SYSLOG system log
/etc/init.d/network start-up script
/etc/config/timeslave.options optional configuration flags
/dev/ttydx tty port attached to clock.

SEE ALSO
chkconfig(1M), inetd(1M), syslog(1M), timed(1M), timedc(1M), icmp(7P)

BUGS
The filtering is not good enough when the local clock is worse than one part
in 10**4 or the network is overloaded.

Timeslave does not communicate its confidence in the time to a timed run-
ning on the local machine.
NAME
uadmin – administrative control

SYNOPSIS
/etc/uadmin cmd fcn

DESCRIPTION
The uadmin command provides control for basic administrative functions. This command is tightly coupled to the System Administration procedures and is not intended for general use. It may be invoked only by the super-user.

The arguments cmd (command) and fcn (function) are converted to integers and passed to the uadmin system call.

SEE ALSO
NAME
uuchec – check the uucp directories and permissions file

SYNOPSIS
/usr/lib/uucp/uuchec [ -v ] [ -x debug_level ]

DESCRIPTION
uuchec checks for the presence of the uucp system required files and directories. Within the uucp makefile, it is executed before the installation takes place. It also checks for some obvious errors in the Permissions file (/usr/lib/uucp/Permissions). When executed with the -v option, it gives a detailed explanation of how the uucp programs will interpret the Permissions file. The -x option is used for debugging. debug-option is a single digit in the range 1-9; the higher the value, the greater the detail.

Note that uuchec can only be used by the super-user or uucp.

FILES
/usr/lib/uucp/Systems
/usr/lib/uucp/Permissions
/usr/lib/uucp/Devices
/usr/lib/uucp/Maxuuscheds
/usr/lib/uucp/Maxuxqts
/usr/spool/uucp/*
/usr/spool/locks/LCK*
/usr/spool/uucppublic/*

SEE ALSO
uucico(1M), uuschec(1M).
uucp(1C), uustat(1C), uux(1C) in the User's Reference Manual.

BUGS
The program does not check file/directory modes or some errors in the Permissions file such as duplicate login or machine name.
NAME
uucico – file transport program for the uucp system

SYNOPSIS
/usr/lib/uucp/uucico [ -r role_number ] [ -x debug_level ]
[ -i interface ] [ -d spool_directory ] -s system_name

DESCRIPTION
uucico is the file transport program for uucp work file transfers. Role
numbers for the -r are the digit 1 for master mode or 0 for slave mode
(default). The -r option should be specified as the digit 1 for master mode
when uucico is started by a program or cron. Uux and uucp both queue jobs
that will be transferred by uucico. It is normally started by the scheduler,
uuchsched , but can be started manually; this is done for debugging. For
example, the shell Uutry starts uucico with debugging turned on. A single
digit must be used for the -x option with higher numbers for more debug-
ging.

The -i option defines the interface used with uucico. This interface only
affects slave mode. Known interfaces are UNIX (default), TLI (basic Trans-
port Layer Interface), and TLIS (Transport Layer Interface with Streams
modules, read/write).

The device names, dialers, and so forth in /usr/lib/uucp/Devices must be
correct. The device names used by uucp, such as /dev/ttyym3, must be read-
able and writable by the "user" uucp.

Any user name under which uucico runs (i.e. any "user" with
/usr/lib/uucp/uucico as its shell in /etc/passwd) must be no more than 8
characters long. This restriction is a result of the nature of the /etc/utmp
file. System-names longer than 8 characters generally do not work because
of UUCP protocol restrictions. In both cases, one tends to suffer strange
"permission problems."

FILES
/usr/lib/uucp/Systems
/usr/lib/uucp/Permissions
/usr/lib/uucp/Devices
/usr/lib/uucp/Devconfig
/usr/lib/uucp/Sysfiles
/usr/lib/uucp/Maxuxuqts
/usr/lib/uucp/Maxuuscheds
/usr/spool/uucp/*
/usr/spool/locks/LCK*
/usr/spool/uucppublic/*
SEE ALSO

cron(1M), uucp(1C), uuschd(1M), uustat(1C), uutry(1M), uux(1C).
NAME

uucleanup — uucp spool directory clean-up

SYNOPSIS

/usr/lib/uucp/uucleanup [ -Ctime ] [ -Wtime ] [ -Xtime ] [ -mstring ]
[ -otime ] [ -s system ]

DESCRIPTION

uucleanup will scan the spool directories for old files and take appropriate
action to remove them in a useful way:

Inform the requestor of send/receive requests for systems that can not be
reached.

Return mail, which cannot be delivered, to the sender.

Delete or execute rnews for rnews type files (depending on where the news
originated—locally or remotely).

Remove all other files.

In addition, there is provision to warn users of requests that have been wait-
ing for a given number of days (default 1). Note that uucleanup will pro-
cess as if all option times were specified to the default values unless time is
specifically set.

The following options are available.

-Ctime   Any C. files greater or equal to time days old will be removed
with appropriate information to the requestor. (default 7 days)

-Dtime   Any D. files greater or equal to time days old will be removed.
An attempt will be made to deliver mail messages and execute
rnews when appropriate. (default 7 days)

-Wtime   Any C. files equal to time days old will cause a mail message to
be sent to the requestor warning about the delay in contacting
the remote. The message includes the JOBD, and in the case
of mail, the mail message. The administrator may include a
message line telling whom to call to check the problem (-m
option). (default 1 day)

-Xtime   Any X. files greater or equal to time days old will be removed.
The D. files are probably not present (if they were, the X. could
get executed). But if there are D. files, they will be taken care
of by D. processing. (default 2 days)
-mstring  This line will be included in the warning message generated by
the -W option.

-o1ime Other files whose age is more than time days will be deleted.
(default 2 days) The default line is "See your local administra-
tor to locate the problem".

-sysystern Execute for system spool directory only.

-xdebug_level
The -x debug level is a single digit between 0 and 9; higher
numbers give more detailed debugging information. (If
uucleanup was compiled with -DSMALL, no debugging output
will be available.)

This program is typically started by the shell uudemon.cleanup, which
should be started by cron(1M).

FILES
/usr/lib/uucp directory with commands used by uucleanup inter-

/usr/spool/uucp spool directory

SEE ALSO
cron(1M).
uucp(1C), uux(1C) in the User's Reference Manual.
NAME
nugetty – set terminal type, modes, speed, and line discipline

SYNOPSIS
/usr/lib/uucp/nugetty [ -h ] [ -t timeout ] [ -r ] [ -d delay ]
               [ -i chat ] [ -D ] line [ speed [ type [ linedisc ] ] ]
/usr/lib/uucp/nugetty -c file

DESCRIPTION
nugetty is similar to getty(1M), but changes have been made to support
using the line for uucico, cu, and ct; that is, the line can be used in both
directions. The nugetty will allow users to login, but if the line is free,
uucico, cu, or ct can use it for dialing out. The implementation depends on
the fact that uucico, cu, and ct create lock files when devices are used.
When the "open()" returns (or the first character is read when -r option is
used), the status of the lock file indicates whether the line is being used by
uucico, cu, ct, or someone trying to login.

The -d option specifies a number of seconds to wait after the first character
is available from the line, and then to discard all input. This option can be
useful with modems which provide 'call progress' information when
answering. For example, the -d option can be used to ignore 'RING' and
'CONNECT' which would otherwise fool nugetty into invoking login
before the correct speed has been determined.

The -i option specifies an entry in /usr/lib/uucp/Dialers which should be
used to initialize the modem. Uugetty uses the chat-script before going to
sleep to wait for the first input. The -D option turns onuucico debugging.
It can only be used when uugetty is invoked manual, and not when it is
invoked from /etc/inittab. It can be very useful for testing a script specified
with the -i option.

Note that when the -r option is used, several <carriage-return> characters
may be required before the login message is output. The human users will
be able to handle this slight inconvenience. Uucico trying to login will have
to be told by using the following login script:

"" \\
in:--in: . . .

where the . . . is whatever would normally be used for the login sequence.

An entry for direct line that has a uugetty on each end must use the -r
option. This causes uugetty to wait to read a character before it puts out the
login message, thus preventing two uugetties from looping. If there is a
uugetty on one end of a direct line, there must be a uugetty on the other end
as well.
Here is an /etc/inittab entry using `uugetty` on an intelligent modem:

```
t4:23:respawn:/usr/lib/uucp/uugetty -r -t 60 -d 8 ttym2 dx_2400
```

Please note that the ttym* or ttyf* device names must be used in the /usr/lib/uucp/Devices and /etc/inittab files and with the `cu` command, because `uugetty` depends on the modem to provide the signal DCD, Carrier Sense, on pin 8, and on the operating system to act on its state. When using the `cu` command on a line shared with `uugetty`, the line cannot be specified explicitly to `cu`. Instead, it must be specified implicitly with the /usr/lib/uucp/Devices and /usr/lib/uucp/Dialers files.

Also note that a /etc/gettydefs entry which cycles among the speed(s) appropriate for the modem should be chosen.

FILES

/etc/gettydefs
/etc/issue

SEE ALSO
duart(7), getty(1M), init(1M), tty(7), uucico(1M).
cu(1C), login(1) in the User’s Reference Manual.

BUGS

`ct` will not work when `uugetty` is used with an intelligent modem such as pensril or ventel.
NAME

tuced – the scheduler for the uucp file transport program

SYNOPSIS

/usr/lib/uucp/uusch[ ] [ -x debug_level ] [ -u debug_level ]

DESCRIPTION

ussched is the uucp file transport scheduler. It is usually started by the dae-
mon uudemon.hour that is started by cron(1M) from an entry in
/usr/spool/cron/crontab:

39 * * * * /bin/su uucp -c "/usr/lib/uucp/uudemon.hour > /dev/null"

The two options are for debugging purposes only; -x debug_level will output debugging messages from uuschd and -u debug_level will be passed as -x debug_level to uucico. The debug_level is a number between 0 and 9; higher numbers give more detailed information.

FILES

/usr/lib/uucp/Systems
/usr/lib/uucp/Permissions
/usr/lib/uucp/Devices
/usr/spool/uucp/*
/usr/spool/locks/LCK*
/usr/spool/uucppublic/*

SEE ALSO

cron(1M), uucico(1M).
uucp(1C), uustat(1C), uux(1C) in the User's Reference Manual.
NAME

Utry – try to contact remote system with debugging on

SYNOPSIS

/usr/lib/uucp/Utry [ -x debug_level ] [ -r ] system_name

DESCRIPTION

Utry is a shell that is used to invoke uucico to call a remote site. Debugging is turned on (default is level 5); -x will override that value. The -r overrides the retry time in /usr/spool/uucp/.status. The debugging output is put in file /tmp/system_name. A tail -f of the output is executed. A <DELETE> or <BREAK> will give control back to the terminal while the uucico continues to run, putting its output in /tmp/system_name.

FILES

/usr/lib/uucp/Systems
/usr/lib/uucp/Permissions
/usr/lib/uucp/Devices
/usr/lib/uucp/Maxuuxqts
/usr/lib/uucp/Maxuuscheds
/usr/spool/uucp/*
/usr/spool/locks/LCK*
/usr/spool/uucppublic/*
/tmp/system_name

SEE ALSO

uucico(1M).

uucp(1C), uux(1C) in the User’s Reference Manual.
NAME
uuxqt – execute remote command requests

SYNOPSIS
/usr/lib/uucp/uuxqt [ -s system ] [ -x debug_level ]

DESCRIPTION
uuxqt is the program that executes remote job requests from remote systems
generated by the use of the uux command. (Mail uses uux for remote mail
requests). uuxqt searches the spool directories looking for X. files. For
each X. file, uuxqt checks to see if all the required data files are available
and accessible, and file commands are permitted for the requesting system.
The Permissions file is used to validate file accessibility and command exe-
cution permission.

There are two environment variables that are set before the uuxqt command
is executed:
UU_MACHINE is the machine that sent the job (the previous one).
UU_USER is the user that sent the job.
These can be used in writing commands that remote systems can execute to
provide information, auditing, or restrictions.

The -x debug_level is a single digit between 0 and 9. Higher numbers give
more detailed debugging information.

FILES
/usr/lib/uucp/Permissions
/usr/lib/uucp/Maxuuxqts
/usr/spool/uucp/*
/usr/spool/locks/LCK*

SEE ALSO
uucico(1M).
uucp(1C), uustat(1C), uux(1C), mail(1) in the User's Reference Manual.
NAME
versions – software versions tool

SYNOPSIS
versions [ options ] [ operator ] [ selectors ]

DESCRIPTION
versions, with no command line options, shows the names of software products and images, with version numbers, installation dates, and installation status. The first column will contain an indication of the installation status of the product, image, or subsystem listed on that line. An “I” indicates that the item is installed, an “R” indicates that it was installed but has been removed, a “C” indicates that the installation history has somehow been corrupted, and a blank indicates that the item has never been installed.

The next column gives the product, image, or subsystem name, followed by the date of installation and/or the version number, and the description of the item.

The operators are used to perform operations on the installed subsystems. In this case, the options and selectors are used to restrict the operations to specific products, images, subsystems, and/or types of files.

The options are as follows:

-m Operate only on modified files; i.e. those that have been altered in some way since they were installed.

-u Operate only on unmodified files; i.e. those that are still as originally installed.

-c Operate only on configuration files, as defined in the software product. (Configuration files are editable text files that may be altered with site or machine-specific configuration information.)

-s Operate only on system (i.e. non-configuration) files.

-S Operate only on diskless client “shared” files; i.e. symbolic links to the /share directory.

-U Operate only on unshared files.

-k Calculate checksums of user files in the long listing.

-i pattern Add a name or filename pattern to be included in user file listings.

-e pattern Add a name or filename pattern to be excluded from the user file listings.
-x  Don’t use the standard exclusions table /usr/lib/inst/userlist.exe for user file listings.

-I  List installed subsystems only; don’t list subsystems that have never been installed, or have been installed and then removed.

-n  In the versions display, show the internal version number of the product or image rather than the date it was installed.

-v  Operate verbosely: include subsystems in the versions list, or display file names if they would otherwise not be, as during a remove.

-p  Use the built-in pausing mechanism, which operates something like more(1).

-r root  Operate on an IRIX tree other than the default, rooted at rootdir. The default root directory while running under IRIX is /, and while in the miniroot is /root. Other root directories might be diskless prototype trees, or test installations that have been done somewhere other than on the running system’s root.

The operators are as follows:

list  List the selected file names.

long  List the selected file names with additional information. This includes the file type, the checksum and size in blocks at time of installation (via ‘‘sum -r’’), the subsystem name, flags, and the file name. The file type is as follows:

  f    Plain file
  d    Directory
  b    Block special
  c    Character special
  l    Symbolic link
  p    Fifo, a.k.a. named pipe

The flags are as follows:

c    Configuration file

t    Temporary (i.e. orphaned) configuration file

m    File is machine specific (see inst(1m))

April 1990  - 2 -  Version 5.0
user

List only user files, i.e., those that have not been installed as part of any software product.

remove

Remove the selected files from the disk.

config

Show configuration file status. This includes the existence of old and new versions saved with .O and .N suffixes, and an indication of whether the various files have been modified since they were installed. This information is useful in reconfiguring a system after a new version of software has been installed. For a file foo, if there is a file foo.O, it is the “old” version of the file, saved during installation so that the new version could be put in place. In this case, the new file was deemed to have precedence for some reason. Changes from the old version can be merged into the new version, if appropriate, and the old version can be removed. If there is a file foo.N, it is the “new” version of the file, installed under a different name so that the old version could remain active. In this case, the new version contains suggested changes that you may wish to use. Once you have extracted all useful information from it, it can be removed.

changed

Show configuration file status, but only where a .O or .N file is present. This operator is quite useful after installing new software, as it isolates the configuration files that were affected by the installation in some way, and need attention.

The selectors are product, image, and/or subsystem names or shell-style patterns. These are used to select specific components of software products. Software products are defined in a three-level hierarchy, the first of which is the product itself. Each product is made of one or more images. Each image is made of subsystems, which are logically-related collections of the individual files of the product. The subsystem is the generally lowest level at which software installation and removal decisions can be made.

Each product, image and subsystem has a short name by which it is known internally, and a longer description giving an indication of its contents. These names are concatenated with dots to select specific components. For example, “foo.bar.main” is the “main” subsystem of the “bar” image of the “foo” software product.

Shell-style patterns may be used to select groups of subsystems; the most common pattern will be of the form “foo.bar.*” to select all subsystems of the named image, or “foo.*.*” to select all subsystems of all images of the named product. Missing components in the selectors are assumed to be “*”, so these last two examples can be abbreviated “foo.bar” and “foo”, respectively.
The special pattern "user" (recognized only for the filename listing operators) can be used to refer to all files on the system that are not part of an installed software product.

FILES

/usr/lib/inst/*  Installation history and supporting files
/usr/lib/inst/userlist.exc  Standard exclusion patterns for user lists

SEE ALSO

inst(1m).
NAME
whodo – who is doing what

SYNOPSIS
/etc/whodo

DESCRIPTION
whodo produces formatted and dated output from information in the /etc/utmp and /tmp/.ps_data files.

The display is headed by the date, time and machine name. For each user logged in, device name, user-id and login time is shown, followed by a list of active processes associated with the user-id. The list includes the device name, process-id, cpu minutes and seconds used, and process name.

EXAMPLE
The command:

    whodo

produces a display like this:

Tue Mar 12 15:48:03 1985
bailey

tty09  mcn  8:51
  tty09  28158  0:29 sh

tty52  bdr  15:23
  tty52  21688  0:05 sh
  tty52  22788  0:01 whodo
  tty52  22017  0:03 vi
  tty52  22549  0:01 sh

xt162  lee  10:20
  tty08  6748  0:01 layers
  xt162  6751  0:01 sh
  xt163  6761  0:05 sh
  tty08  6536  0:05 sh

FILES
/etc/passwd
/tmp/.ps_data
/etc/utmp
SEE ALSO
NAME
intro – introduction to special files

DESCRIPTION
This section describes various special files that refer to specific hardware peripherals, and UNIX system device drivers. STREAMS [see intro(2)] software drivers, modules and the STREAMS-generic set of ioctl(2) system calls are also described.

For hardware related files, the names of the entries are generally derived from names for the hardware, as opposed to the names of the special files themselves. Characteristics of both the hardware device and the corresponding UNIX system device driver are discussed where applicable.

Disk device file names are in the following format:

/dev/{r}dsk/<device-type><controller-#>d<drive-#>{s<slice-#>|vh|vol}

where r indicates a raw interface to the disk, the <device-type> indicates the type of device, <controller-#> indicates the controller number, <drive-#> indicates the device attached to the controller (drive number) and <slice-#> indicates the section number of the partitioned device. Certain sections have alternate names, namely vh for the volume header and vol for the entire volume: header, defects and all.

Tape device file names are in the following format:

/dev/{r}mt/<device-type><controller-#>d<slave-#>{nr}{.density}

where r indicates a raw interface to the disk, the <device-type> identifies the type of device that the tape is, <controller-#> indicates the controller number, <slave-#> indicates the device attached to the controller (slave number), nr indicates a non rewinding interface, and density optionally specifies the media density, where appropriate. For devices with only one density setting, density may be omitted. The "." is used to keep the <slave-#> visually merging with the density.

SEE ALSO
prtvtoc(1M)
Disk/Tape Management in the System Administrator's Guide.
NAME

arp – Address Resolution Protocol

DESCRIPTION

ARP is a protocol used to dynamically map between DARPA Internet and 10Mb/s Ethernet addresses. It is used by all the 10Mb/s Ethernet interface drivers. It is not specific to Internet protocols or to 10Mb/s Ethernet, but this implementation currently supports only that combination.

ARP caches Internet-Ethernet address mappings. When an interface requests a mapping for an address not in the cache, ARP queues the message which requires the mapping and broadcasts a message on the associated network requesting the address mapping. If a response is provided, the new mapping is cached and any pending message is transmitted. ARP will queue at most one packet while waiting for a mapping request to be responded to; only the most recently "transmitted" packet is kept.

To facilitate communications with systems which do not use ARP, ioctls are provided to enter and delete entries in the Internet-to-Ethernet tables. Usage:

```c
#include <sys/ioctl.h>
#include <sys/socket.h>
#include <net/if.h>
struct arpreq arpreq;

ioctl(s, SIOCSARP, (caddr_t *)&arpreq);
ioctl(s, SIOCGARP, (caddr_t *)&arpreq);
ioctl(s, SIOCDARP, (caddr_t *)&arpreq);
```

Each ioctl takes the same structure as an argument. SIOCSARP sets an ARP entry, SIOCGARP gets an ARP entry, and SIOCDARP deletes an ARP entry. These ioctls may be applied to any socket descriptor s, but only by the super-user. The arpreq structure contains:

```c
/* ARP ioctl request */
struct arpreq {
    struct sockaddr *arp_pa;  /* protocol address */
    struct sockaddr *arp_ha;  /* hardware address */
    int arp_flags;  /* flags */
};
/* arp_flags field values */
#define ATF_COM 0x02  /* completed entry (arp_ha valid) */
#define ATF_PERM 0x04  /* permanent entry */
#define ATF_PUBL 0x08  /* publish (respond for other host) */
#define ATF_USETRAILERS 0x10 /* send trailer packets to host */
```
The address family for the _arp_pa_ sockaddr must be AF_INET; for the _arp_ha_ sockaddr it must be AF_UNSPEC. The only flag bits which may be written are ATF_PERM, ATF_PUBL and ATF_USETRAILERS. ATF_PERM causes the entry to be permanent if the ioctl call succeeds. The peculiar nature of the ARP tables may cause the ioctl to fail if more than 8 (permanent) Internet host addresses hash to the same slot. ATF_PUBL specifies that the ARP code should respond to ARP requests for the indicated host coming from other machines. This allows a host to act as an "ARP server," which may be useful in convincing an ARP-only machine to talk to a non-ARP machine.

ARP is also used to negotiate the use of trailer IP encapsulations; trailers are an alternate encapsulation used to allow efficient packet alignment for large packets despite variable-sized headers. Hosts which wish to receive trailer encapsulations so indicate by sending gratuitous ARP translation replies along with replies to IP requests; they are also sent in reply to IP translation replies. The negotiation is thus fully symmetrical, in that either or both hosts may request trailers. The ATF_USETRAILERS flag is used to record the receipt of such a reply, and enables the transmission of trailer packets to that host.

ARP watches passively for hosts impersonating the local host (i.e. a host which responds to an ARP mapping request for the local host’s address).

**DIAGNOSTICS**

The following messages can appear on the console:

arp: host with ether address %x:%x:%x:%x:%x:%x is using my IP address x.x.x.x

ARP has discovered another host on the local network which responds to mapping requests for its own Internet address.

arp: ether address is broadcast for IP address x.x.x.x

ARP has discovered another host on the local network which maps that host’s IP address onto the ethernet broadcast address.

**SEE ALSO**

inet(7F), arp(1M), ifconfig(1M), intro(3)

"An Ethernet Address Resolution Protocol," RFC826, Dave Plummer, Network Information Center, SRI.

"Trailer Encapsulations," RFC893, S.J. Leffler and M.J. Karels, Network Information Center, SRI.

**BUGS**

ARP packets on the Ethernet use only 42 bytes of data; however, the smallest legal Ethernet packet is 60 bytes (not including CRC). Some systems may not enforce the minimum packet size, others will.
NAME

audio – bi-directional audio channel interface

SYNOPSIS

/dev/audio

DESCRIPTION

The special file /dev/audio refers to the bi-directional audio interface on the IRIS 4D/20. The audio device is capable of digital record and playback as well as sound synthesis. The device consists of a bi-directional, 8-bit D/A and A/D converter with software adjustable output gain. Three different sampling frequency are available: 33khz, 16khz, and 8khz.

The software interface to the audio channel consists of a single device, /dev/audio, that supports direct reads and writes to the D/A and A/D converters. Output gain and sampling rate may be controlled via ioctl calls to the device. A read of the audio channel will fill a buffer with eight bit samples taken at the current frequency until the size of the read request is satisfied. A write to the audio channel will send a buffer one byte at a time at the current frequency until the buffer is exhausted. For sound synthesis, a sound duration may be specified in 1/100 Hz increments. If a duration is specified and the buffer is exhausted before the elapsed time has passed, the buffer is replayed again from the beginning.

The following ioctls are defined in the include file <sys/audio.h> and can be used to control the audio channel from within a program:

AUDIOCSETOUTGAIN

Set the output gain control to the value specified by the argument. Valid values for output gain are numbers in the range 0 to 255.

AUDIOCGETOUTGAIN

Returns the current setting of the audio output gain.

AUDIOCSETRATE

Sets the input and output sampling rate according to the value of the argument. Possible settings are: 1=33khz, 2=16khz, and 3=8khz.

AUDIOCDURATION

Sets the sound duration to the value (in 1/100 Hz units) specified by the argument. When a duration is set, recording or playback continues until the duration time expires. If the i/o request length of the is exceeded
before time is up, the buffer is re-used. This feature can be used to continuously repeat a waveform for sound synthesis.

FILES
/dev/audio
NAME
cdio – 6-port serial I/O

SYNOPSIS
/dev/tty[dmf][5-10]
/dev/tty[dmf][13-18]

DESCRIPTION
6-port boards can be used for modems or other asynchronous, serial I/O
devices. The ports provided by the cdio board are very similar to the on-
board duart ports.

Various character echoing and interpreting parameters can be set indepen-
dently for each port. See termio(7) for details.

By opening either the ttyd*, ttym*, or ttyf* device of a port, different
hardware signals are supported. See the chapter "Terminals, Modems, and
Dumb Printers" in the IRIS–4D System Administrator’s Guide for detailed
information on supported signals. Typically, ttyd* is used for direct con-
nect devices such as most terminals; ttym* is used for devices that use
modem control such as modems; and ttyf* is used for devices that under-
stand hardware flow control such as most printers. If ttyf* is desired, the
"new" cabling scheme be used. The driver is informed of which cabling is
present by its lboot(1M) file, /usr/sysgen/master.d/cdio.

To allow the system and the board to handle high volume, high speed input,
individual characters are accumulated for short periods before they are
given to the rest of the system. While this does make higher throughput
possible, it also increases the latency for individual characters. This latency
can be adjusted in the the lboot(1M) file.

Detection of the serial I/O board is automatic, and if the board is present, its
driver is included in the configured kernel. This probing and configuring is
done when the system starts by the rc2 scripts.

FILES
/dev/tty[dmf][5-10]
/dev/tty[dmf][13-18]
/dev/MAKEDEV
/etc/init.d/autoconfig
/usr/sysgen/master.d/cdio
/usr/sysgen/system

SEE ALSO
duart(7), lboot(1M), rc2(1M), system(4), termio(7)
NAME
clone – open any minor device on a STREAMS driver

DESCRIPTION

clone is a STREAMS software driver that finds and opens an unused minor device on another STREAMS driver. The minor device passed to clone during the open is interpreted as the major device number of another STREAMS driver for which an unused minor device is to be obtained. Each such open results in a separate stream to a previously unused minor device.

The clone driver consists solely of an open function. This open function performs all of the necessary work so that subsequent system calls (including close(2)) require no further involvement of clone.

clone will generate an ENXIO error, without opening the device, if the minor device number provided does not correspond to a valid major device, or if the driver indicated is not a STREAMS driver.

CAVEATS

Multiple opens of the same minor device cannot be done through the clone interface. Executing stat(2) on the file system node for a cloned device yields a different result from executing fstat(2) using a file descriptor obtained from opening the node.

SEE ALSO

log(7).
NAME
console – console interface

DESCRIPTION
The console provides the operator interface to the system. The operating system and system utility programs display error messages on the system console.

The console can be a serial terminal connected to the first serial port on the back panel of the workstation or it can be a logical terminal represented by a text window on the graphics monitor.

The device special file /dev/console represents the system console. When the console is a window on the graphics monitor, /dev/console will be the slave side of pseudo-tty (see pty(7)).

When the console is a serial terminal, the file /dev/console will be the appropriate DUART device.

FILES
/dev/console

SEE ALSO
termio(7).
NAME
dks – Small Computer Systems Interface (SCSI) disk driver

SYNOPSIS
/dev/dsk/dks*
/dev/rdsk/dks*

DESCRIPTION
There may be up to 7 SCSI drives attached to a system. The special files are named according to the convention discussed in intro(7M):

/dev/[r]dsk/dks<controller-#>/d<drive-#>{s<partition-#>}[vh|vol]

The standard partition allocation by Silicon Graphics has root on partition 0, swap on partition 1, and /usr on partition 6. Partition 7 maps the entire usable portion of the disk (excluding the volume header). Partition 8 maps the volume header (see privtoc(1M), dvhtool(1M)). Partition 10 maps the entire drive. The remaining partitions are reserved for use by Silicon Graphics.

The standard configuration has /dev/root linked to /dev/dsk/dks0d1s0, /dev/swap linked to /dev/dsk/dks0d1s1, and /dev/usr linked to /dev/dsk/dks0d1s6, for systems with the root on a SCSI drive.

IOCTL FUNCTIONS
As well as normal read and write operations, the driver supports a number of special ioctl(2) operations when opened via the character special file in /dev/rdsk. Command values for these are defined in the system include file <sys/dkio.h>, with data structures in <sys/dksc.h>.

These ioctl operations are intended for the use of special-purpose disk utilities. Many of them can have drastic or even fatal effects on disk operation if misused; they should be invoked only by the knowledgeable and with extreme caution!

A list of the ioctl commands supported by the dks driver is given below.

DIOCADDDBB
adds a block to the badblock list. The argument is the logical block number (not a pointer) on the drive. For some makes of drives, the spared block must be written before the sparing takes effect. Only programs running with superuser permissions may use this ioctl.

DIOCDRVTYPE
The first 24 bytes of the SCSI inquiry data for the drive is returned to the caller. The argument is a pointer to a char array of at least 24 bytes. This contains vendor and drive specific information such as the drive name and model number. See a SCSI command
specification for details on the structure of this buffer.

**DIOCFORMAT**

formats the entire drive. Any information on the drive is lost. The
grown defect list (blocks spared with DIOCADDDBB) is empty
after formatting is complete, blocks previously in the grown defect
list are no longer spared.

**DIOCGETVH**

reads the disk volume header from the driver into a buffer in the
calling program. The argument in the ioctl call must point to a
buffer of size at least 512 bytes. The structure of the volume
header is defined in the include file `<sys/dvh.h>`. The correspond-
ing call DIOCSETVH sets the drivers idea of the volume header;
in particular, the drivers idea of the partition sizes and offsets is
changed.

**DIOCRDEFFECTS**

The argument is a pointer to a 'struct dk_ioctl_data’. The i_addr
field field points to a structure like:

```c
structure defect_list {
    struct defect_header defhdr;
    struct defect_entry defentry[NENTS];
};
```

the i_len field is set to the total length of the structure, which must
be less than NBPP from `<sys/param.h'>; at most NENTS defects
will be returned. The actual number of defects may be determined
by examining the defhdr.defect_listlen value, which is the number
of bytes returned. This must be divided by the size of the applica-
ble data structure for the type requested. The i_page field should
be set to the bits identifying the badblock reporting type. These
bits request the combination of manufacturer's and grown defects;
and one of bytes from index, physical cyl/head/sec, vendor unique.
The only combination that works with all currently supported SCSI
disks is type cyl/head/sec; and either combined manufacturer’s and
grown defects, or just manufacturer’s defects.

**DIOCREADCAPACITY**

The arg is a pointer to an unsigned integer. The value returned is
the number of usable sectors on the drive (as read from the drive).

**DIOCSENSE / DIOCSELECT**

The argument is a pointer to a 'struct dk_ioctl_data’. This allows
sending SELECT and SENSE commands to the drive. See the
ANSI SCSI specification and individual manufacturer’s manuals
for allowed page numbers and valid values. Only programs
running with superuser permissions may use the DIOCSELECT ioctl.

DIOCTEST
issues the SCSI "Send Diagnostic" command to the drive. The exact tests done are manufacturer specific. The ioctl call will return 0 upon success, or will set errno to EIO and return -1 upon failure.

FILES
/dev/dsk/dks*, /dev/rdsk/dks*
/dev/root, /dev/usr, /dev/swap

SEE ALSO
dvhtool(1M), prtvtoc(1M), fx(1M).

NOTE
As of the IRIX 3.2 release, the driver attempts to negotiate synchronous SCSI mode with the drive when it is opened. When supported by the drive, this results in greater disk throughput, and better SCSI bus utilization when multiple devices are attached to the SCSI bus. If problems occur because of this (due to use of unsupported drives that don’t properly handle this negotiation), you may disable this negotiation by changing the scsi_syncenable variable in the file /usr/sysgen/master.d/scsi as described by the comments in that file, and linking a new kernel with lboot(7M). In the IRIX 3.3 release, the kernel will automatically disable the synchronous SCSI mode for systems that don’t support this feature regardless of what is set in scsi_syncenable.

Another variable of possible interest in /usr/sysgen/master.d/scsi is scsi_enable_disconnect If this variable is set to 0, the SCSI host adapter will be programmed to NOT support device disconnects. This can be useful with devices that don’t support disconnect which might otherwise cause SCSI bus timeouts. It is also useful with devices which purport to support disconnect, but which don’t work correctly when it is enabled.
NAME
dn_ll, dn_netman – 4DDN special files

DESCRIPTION
The dn_ll special file is used to create 4DDN logical links. Logical links are temporary software data paths established between two communicating processes in a 4DDN network and use DECnet phase-IV protocols.

The dn_netman special file is used by the 4DDN network management software.

FILES
/dev/dn_ll
/dev/dn_netman

SEE ALSO

NOTE
These files are used only with the optional 4DDN software. DECnet is a trademark of Digital Equipment Corporation.
NAME
drain – capture unimplemented link-layer protocols

SYNOPSIS
#include <sys/types.h>
#include <net/raw.h>

s = socket(PF_RAW, SOCK_RAW, RAWPROTO_DRAIN);

DESCRIPTION
The Drain protocol provides non-promiscuous capture of packets having
unimplemented link-layer protocol types, i.e., packets that the operating
system normally receives and drops “down the drain”. It treats packets as
datagrams containing a link-layer header followed by data. Drain uses the
Raw address format, interpreting ports as link-layer type codes (in host byte
order) to match against unimplemented types in received packets. Multiple
sockets may bind to the same port on a network interface.

Drain can map several link-layer type codes to a port. There is one type-
to-port mapping for each network interface; it is initialized to map zero to
zero. Call ioctl(2) with the SIOCDRAINMAP command and the address of
the following structure to set a mapping:

```
struct drainmap {
    u_short dm_minport; /* lowest port in range */
    u_short dm_maxport; /* and highest port */
    u_short dm_topoport; /* port mapped by range */
};
```

Drain input from Ethernet network interfaces is demultiplexed based on the
ether_type member of the ether_header structure, which is declared in
<netinet/ether.h>.

If the link-layer header size is not congruent with RAW_ALIGNGRAIN,
Drain input prepends RAW_HDRPAD(hdrsize) bytes of padding to received
packets. Output on a Drain socket, using write(2) or send(2), takes a buffer
address pointing at the link-layer packet to be transmitted, not at any
prepended padding.

EXAMPLES
To capture from an Ethernet network interface, first declare an input buffer
structure with the required header padding:
#include <sys/types.h>
#include <net/raw.h>
#include <netinet/if_ether.h>

#define ETHERHDRPAD RAW_HDRPAD(sizeof(struct ether_header))

struct etherpacket {
    char pad[ETHERHDRPAD];
    struct ether_header ether;
    char data[ETHERMTU];
}

To capture all Reverse ARP (RARP) packets, create a Drain socket and
bind it to the RARP port on the primary network interface (error handling is
omitted for clarity):

#define ETHERTYPE_RARP 0x8035

int s;
struct sockaddr_raw sr;

s = socket(PF_RAW, SOCK_RAW, RAWPROTO_DRAIN);
bzero(sr.nr_ifname, sizeof sr.nr_ifname);
sr.nr_port = ETHERTYPE_RARP;
bind(s, &sr, sizeof sr);

Alternatively, to capture all Ethernet packets with IEEE 802.3 encapsu-
lations, create and bind a socket to a port different from any valid ether_type:

#define IEEE802_3PORT 1

int s;
struct sockaddr_raw sr;

s = socket(PF_RAW, SOCK_RAW, RAWPROTO_DRAIN);
bzero(sr.nr_ifname, sizeof sr.nr_ifname);
sr.nr_port = IEEE802_3PORT;
bind(s, &sr, sizeof sr);

Map all Ethernet types corresponding to packet lengths, as specified by
802.3, to the bound port:
struct drainmap map;

map.dm_minport = 60;
map.dm_maxport = 1514;
map.dm_toport = IEEE802_3PORT;
ioctl(s, SIOC DRAINMAP, &map);

Before reading, it may be desirable to increase the Drain socket’s default receive buffer size. The following code also shows how to transmit a link-layer packet:

struct etherpacket ep;
int cc = 10000;

setsockopt(s, SOL_SOCKET, SO_RCVBUF, (char *)&cc, sizeof cc);
for (;;) {
    cc = read(s, (char *)&ep, sizeof ep);
    /* . . . */
    write(s, (char *)&ep.ether, cc - sizeof ep.pad);
}

DIAGNOSTICS
A socket operation may fail with one of the following errors returned:

[EISCONN] when trying to establish a connection on a socket which already has one, or when trying to send a datagram with the destination address specified and the socket is already connected;

[ENOBUFS] when the system runs out of memory for an internal data structure or a send or receive buffer.

[EADDRINUSE] when an attempt is made to create a socket with a port which has already been allocated;

[EADDRNOTAVAIL] when an attempt is made to create a socket with a network address for which no network interface exists.

[EOPNOTSUPP] when an ioctl operation not supported by the Drain protocol is attempted.

SEE ALSO
getssockopt(2), socket(2), intro(3), ethernet(7), raw(7P), snoop(7P)
NAME
ds – generic (user mode) SCSI driver

SYNOPSIS
/dev/scsi/sc0*

DESCRIPTION
The ds interface provides user-mode access to the SCSI bus. It supports the
programming interfaces described in dslib(3) and below.

This interface is under active development, and a variation is under con-
sideration by the SCSI-2 Common Access Method (CAM) Committee
Unix/OSD working group. The interface may change somewhat to conform
to the standard, if it is adopted.

The driver uses one UNIX major device number for each SCSI bus (host
adapter) supported. Currently the driver only supports one SCSI host
adapter. The minor number is then used to specify ID’s (iii), LUN’s (lll),
and implementation specific control bits (cc). The layout of the minor dev-
vice number is ccllliii. This allows a device to have the same value for its
SCSI ID and minor device number, unless LUN or control bits are needed.

All of the ds devices support the same general interface. The program calls
open, gaining exclusive use of a specified SCSI device. The driver uses the
major and minor numbers of the inode to determine the BUS, ID, and LUN,
along with any implementation-dependent control bits. Subsequent ioctl
calls are either performed directly or turned into SCSI transactions. Finally,
the close command releases the device for further use.

The critical interface definitions (request structure and codes, return codes,
etc.) are defined in <sys/dsreq.h>. In particular, most SCSI transactions
are performed using the dsreq structure:

typedef struct dsreq {  /* DEVSCSI ioctl structure */
  ulong ds_flags;       /* devscsi prefix .............. */
  ulong ds_time;        /* see DSRQ_* flags below */
  ulong ds_private;     /* timeout in milliseconds */
                        /* (1 Hz used if zero) */
  caddr_t ds_cmdbuf;    /* private use by caller */
  uchar_t ds_cmdlen;    /* SCSI request ............... */
  caddr_t ds_databuf;   /* command buffer */
  ulong ds_datalen;     /* command buffer length */
  caddr_t ds_sensebuf;  /* data buffer start */
  uchar_t ds_senseilen; /* total data length */
  caddr_t ds_senseelen; /* sense buffer */
  uchar_t ds_senseelen; /* sense buffer length */
  /* scatter-gather, etc. ... */

April 1990 - 1 - Version 5.0
dsiovec_t *ds_iovbuf;  /* scatter-gather dma control  */
ushort ds_iovlen;      /* length of scatter-gather    */
struct dsreq *ds_link; /* for linked requests        */
ushort ds_synch;       /* synchronous transfer control */
uchar_t ds_revcode;    /* devscsi version code       */

  /* return portion ............ */
uchar_t ds_ret;        /* devscsi return code        */
uchar_t ds_status;     /* status byte value           */
uchar_t ds_msg;        /* message byte value          */
uchar_t ds_cmdsent;    /* actual length command       */
ulong ds_datsent;      /* actual length user data     */
uchar_t ds_sensesent;  /* actual length sense data    */
} dsreq_t;

The first two parts of the structure (devscsi prefix, SCSI request) must be filled in by the calling program. The third part (scatter-gather, etc.) is largely optional, and the last part (return portion) is used only for returned information.

Normally, the ds_data* fields are used to control data transmission. In this mode, all sent (received) data uses a single I/O buffer. If desired, however, the ds_iov* fields may be used (see DSRQ_IOV) to support a set of I/O buffers. The number of scatter gather entries supported is given by the V_MAX define, and is currently 1. The scatter-gather structure, dsiovec_t, is defined as follows:

typedef struct dsiovec { /* DEVSCSI scatter-gather control */
caddr_t iov_base;     /* i/o base */
  long iov_len;        /* i/o length */
} dsiovec_t;

Different /dev/scsi implementations will support different subsets of the specification. Items in the following tables are therefore marked to indicate the likelihood of support:

! required (must be supported)
. normal (usually supported)
? unusual (seldom supported)

IOCTLS, ETC.

Several ioctl's are supported by /dev/scsi:

    DS_ENTER    struct dsreq !    enter a request
    DS_CANCEL   struct dsreq ?    cancel a request
    DS_POLL     struct dsreq ?    sample a request
DS_CONTIN  struct dsreq  ?  continue a request
DS_RESET  ulong  .  reset SCSI bus
DS_SET  ulong  !  set permanent flags
DS_CLEAR  ulong  !  clear permanent flags
DS_GET  ulong  !  get permanent flags
DS_CONF  struct dsconf  !  get configuration data

The DS_ENTER ioctl is the basis for most interaction with the driver. The user program prepares a request structure, issues the ioctl, and examines the returned status information.

Other ioctls help to fill out the interface, however. The polled I/O ioctls (DS_CANCEL, DS_POLLE, DS_CONTIN) support asynchronous /dev/scsi operations. The bus reset ioctl, if provided, allows a suitably privileged program to reset the SCSI bus.

The permanent flag ioctls (DS_SET, DS_GET, DS_CLEAR) allow access to internal driver flag bits. These are undefined, implementation specific, and should be avoided if portable code is desired.

The DS_CONF ioctl, by contrast, allows a user program to detect (and perhaps handle) implementation-specific configuration parameters:

```c
typedef struct dsconf { /* DEVSCSI configuration structure */
    ulong dsc_flags; /* DSRQ flags honored by driver */
    ulong dsc_preset; /* DSRQ flag preset values */
    ulong dsc_bus; /* # of this SCSI bus */
    ulong dsc_imax; /* # of ID’s per bus */
    ulong dsc_lmax; /* # of LUN’s per ID */
    ulong dsc_iomax; /* maximum length of any I/O’s */
    ulong dsc_biomax; /* maximum length of buffered I/O’s */
} dsconf_t;
```

Most of the dsconf members (dsc_bus, dsc_*max) have obvious meanings. The dsc_flags and dsc_preset words, however, require a bit of explanation. They work together to indicate how the driver will interpret the DSRQ_* flag bits.

These bits are ORed by the driver with the ds_flags word in dsreq, request specific driver actions. The implementation is then free to reject, honor, or ignore them. Specifically, options will not be turned off, but may be rejected via the DSRT_UNIMPL return code. Options may be turned on without any notice whatsoever.
The *dsc_flags* member of *dsconf* indicates which flags the implementation promises to honor. The *dsc_preset* word indicates, for each flag not honored, the value defined by the implementation. By appropriate logical operations, an application may determine which DSRQ_* options are actually available. The action in parentheses is taken when the flag is not set.

**devssci options:**

- **DSRQ_ASYNC**: ? no (yes) sleep until request done
- **DSRQ_SENSE**: . yes (no) auto get sense on status
- **DSRQ_TARGET**: ? target (initiator) role

**select options:**

- **DSRQ_SELATN**: . select with (without) atn
- **DSRQ_DISC**: . identify disconnect (not) allowed
- **DSRQ_SYNXFR**: . (no) use SCSI synchronous transfer
- **DSRQ_SELMSG**: . send supplied (generate) message

**data transfer options:**

- **DSRQ_IOV**: . scatter-gather (not) specified
- **DSRQ_READ**: ! input from SCSI bus
- **DSRQ_WRITE**: ! output to SCSI bus
- **DSRQ_BUF**: . buffered (direct) data transfer

**progress/continuation callbacks:**

- **DSRQ_CALL**: ? notify progress (completion-only)
- **DSRQ_ACKH**: ? (don't) hold ACK asserted
- **DSRQ_ATNH**: ? (don't) hold ATN asserted
- **DSRQ_ABORT**: ? abort msg until bus clear

**host options (non-portable):**

- **DSRQ_TRACE**: . trace (no) this request
- **DSRQ_PRINT**: . print (no) this request
- **DSRQ_CTRL1**: . request with host control bit 1
- **DSRQ_CTRL2**: . enable driver debug printfs during ioctl
                   (if program has superuser privileges)

**additional flags:**
DSRQ_MIXRDWR ? request can both read and write

RETURN CODES

The driver has a number of possible return codes, signifying failures on the part of the driver, the host SCSI software, or the protocol. Some return codes may be mapped to more generic return codes (DSRT_DEVSCSI, DSRT_HOST, DSRTPROTO). Note that the ds_status field contains valid completion status only when the command completed 'normally'. On timeouts and some other errors, this field is set to 0xff on return to indicate that is not valid. The ds_ret may be non-zero even if the command completed successfully; i.e. on partial i/o completion, and when a request sense has been done.

devscsi software returns:

DSRT_DEVSCSI ! general devscsi failure
DSRT_MULTI ! request rejected
DSRT_CANCEL ! lower request cancelled
DSRT_REV_CODE ! software obsolete, must recompile
DSRT_AGAIN ! try again, recoverable bus error
DSRT_UNIMPL ! unimplemented request option

host SCSI software returns:

DSRT_HOST ! general host failure
DSRT_NOSEL . no unit responded to select
DSRT_SHORT ! incomplete transfer (not an error)
DSRT_OK ! completetransfer (no error status)
DSRT_SENSE ! cmd w/ status, sense data gotten
DSRT_NOSENSE ! cmd w/ status, error getting sense
DSRT_TIMEOUT ! request idle longer than requested
DSRT_LONG ! target overran data bounds

protocol failures:

DSRT_PROTO ! misc. protocol failure
DSRT_EBSY . busy dropped unexpectedly
DSRT_REJECT . message reject
DSRT_PARITY . parity error on SCSI bus
DSRT_MEMORY . host memory error
DSRT_CMDO . error during command phase
DSRT_STAI . error during statusphase
SUPPORT CODE

A number of ancillary macros, functions, and data structures are defined in <dslib.h>. While not strictly necessary, these should facilitate the task of programming SCSI control programs.

ADDITIONAL INFORMATION

Consult the SCSI standards documents, and the manuals for the device you are working with for more information. The "SCSI 1" specification document is called SCSI Specification, ANSI X3T9.2/86-109. Also of interest is the Common Command Set specification document SCSI CCS Specification, ANSI X3T9.2/85-3

NOTES

The ds programming interface contains a number of optional features. The control program must therefore be able to react properly, should a desired function be unavailable.

The peculiarities of any given SCSI device are the responsibility of the control program. The /dev/scsi interface merely allows communication to take place.

Since the driver provides direct access to the SCSI bus, the system administrator must be very careful in setting up the permissions on the device files, lest security holes occur.

No kernel read/write interface is provided, due to the variety of forms these commands take in terms of both size and field definitions.

No real test has been made of the multiple bus support code.

No support currently exists for target mode or asynchronous (polled) I/O.

No checking is currently performed for potentially dangerous actions (Copy, ID and code downloading, etc.), except that a program must have super user privileges to use the DS_RESET ioctl.

FILES

/dev/scsi/sc0d[1-7]/l[0-7]

SEE ALSO

dslib(3) for a discussion of routines to simply use of the driver.
dksc(7M) has a NOTES section describing some configuration options of the underlying SCSI driver.
NAME
duart — on-board serial ports

SYNOPSIS
/dev/tty[dmf][1-4,45-56]

DESCRIPTION
Each IRIS-4D machine uses a DUART to connect the mouse and the keyboard. All models are also equipped with some number of additional ‘on-board’ serial ports. The Personal Iris has two ports, each of the Professional Iris Series have four ports, and the Power Series has up to sixteen ports. The first of these ports, /dev/ttyd1, is often used for the ‘serial’ or ‘debugging’ console. Other ports are commonly used for serial terminal connections, modems, a dial-and-button box, or a bit-pad.

Special files for the serial ports exist in the /dev directory. These files, tty[dfm][1-4] are created by MAKEDEV(1M). On multiprocessor systems, four ports exist for each CPU board. The first set of ports, on the first CPU board, is tty[dfm][1-4] as on all other 4D machines. If a second CPU board is installed, these ports will be tty[dfm][45-48]; a third gives tty[dfm][49-52]; and a fourth tty[dfm][53-56].

Each line may independently be set to run at any of several speeds, as high as 19,200 or even 38,400 bps. Various character echoing and interpreting parameters can also be set. See termio(7) for details.

By opening either the ttyd*, ttym*, or ttyf* device of a port, different hardware signals are supported. See the chapter "Terminals, Modems, and Dumb Printers" in the IRIS-4D System Administrator's Guide for detailed information on supported signals. Typically, ttyd* is used for direct connect devices such as most terminals; ttym* is used for devices that use modem control such as modems; and ttyf* is used for devices that understand hardware flow control such as most printers.

The driver for this device must be configured in the kernel by specifying it with ‘INCLUDE’ in the /usr/sysgen/system file.

FILES
/dev/tty[dmf][1-4,45-56]
/dev/MAKEDEV
/usr/sysgen/system

SEE ALSO
cdsio(7), termio(7), system(4)
NAME

ethernet – IRIS-4D Series ethernet controllers

DESCRIPTION

The IRIS-4D Series supports local-area networking with Ethernet. The Ethernet protocol is supported with a hardware controller and a kernel driver. Though the controllers are different among IRIS-4D’s, their drivers provide the same programming interface to networking routines.

Each IRIS ethernet controller is named using the following convention: the prefix is ‘e’ and the suffix is the controller unit number.

<table>
<thead>
<tr>
<th>Controller name</th>
<th>Type</th>
<th>Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>ec0</td>
<td>on-board</td>
<td>Personal Iris, Data Station</td>
</tr>
<tr>
<td>et0</td>
<td>on-board</td>
<td>POWER Series</td>
</tr>
<tr>
<td>enp0, enp1,...</td>
<td>CMC ENP-10</td>
<td>all 4D's</td>
</tr>
</tbody>
</table>

Depending on the model, an IRIS-4D can support several ethernet controllers, allowing it to act as a gateway among different local networks.

The ethernet boards are initialized during system startup from /etc/init.d/network (see network(1M) for details).

The IRIS-4D’s use the 10Mbit/sec Ethernet encapsulation format. Each packet has a 14-byte header, as defined in the #include file <netinet/if_ether.h>:

```c
struct ether_header {
    u_char ether_dhost[6]; /* destination address */
    u_char ether_shost[6]; /* source address */
    u_short ether_type;    /* packet type */
};
```

The packet type determines which kernel protocol routine is called to process the packet data. Examples of common packet types are IP, ARP, and DECnet.

DIAGNOSTICS

Various error messages are printed by the kernel when a problem is encountered. The message is preceded by the controller name, e.g., enp0. Serious errors are flagged with a dagger (†). If they occur repeatedly, contact your product support service for assistance.

The following error messages are common to all controllers:

packet too small (length = X)
packet too large (length = X)
The controller received a packet that was smaller than the minimum ethernet packet size of 60 bytes or larger than the maximum of 1514. This problem is caused by another machine with a bad ethernet controller or transceiver.

stray interrupt
early interrupt
The controller interrupted the kernel before the device was initialized. This error is innocuous; it occurs after booting a kernel over the network from the PROM monitor.

died and restarted
The controller failed to respond after a certain amount of time and the driver had to reset it.

cannot handle address family
This message indicates an error in the kernel protocol handling routines.†

The following messages are specific to the CMC ENP-10 controller:

missing
The controller did not respond to a kernel probe. This message is expected if the controller is not installed in the machine.

operation not supported by the firmware
The kernel tried to change the multicast address filter or the ethernet address or go into promiscuous mode, but failed because the firmware is out-of-date. Use the hinv(1M) command to determine the firmware version and contact your product support service for information on upgrading the firmware.

firmware failed to start

detected error on reset
timed-out waiting for reset
These messages indicates that the controller did not reset properly.†

address command failed, resetting board
An operation that tried to change the controller mode failed, probably because the controller firmware failed. The kernel had to reset the board.†

The following messages are specific to the ec and et controllers.

transmit: no carrier
Indicates the ethernet cable is unplugged from the machine.
late collision
The controller tried to transmit a packet but received a late collision signal from another machine. Usually indicates a problem in the ethernet cable layout.

transmit buffer error
receive buffer error
transmit underflow
receive packet overflow
The controller ran out of memory when trying to transmit or receive a packet.†

unknown interrupt
The controller interrupted the kernel but the reason for the interrupt is missing.†

babbling
The kernel tried to transmit a packet larger than the maximum size.†

machine has bad ethernet address: x:x:x:x:x:x
The ethernet address obtained from non-volatile RAM during controller initialization was corrupted.†

memory timeout
The LANCE ethernet chip failed to access its local memory.†

Counts of ethernet input and output errors can be displayed with the command netstat -i (see netstat(1M)). Typically, output errors and collisions occur due to mismatched controller and transceiver configurations. Input error statistics include counts of the errors listed above and counts of protocol input queue overflows.

SEE ALSO
netstat(1M), network(1M), socket(2), ip(7P), raw(7P), snoop(7P), drain(7P)

NOTE
IEEE 802.3 Ethernet encapsulation is not currently supported. The IRIS-4D ethernet controllers will support IEEE 802.3 and Ethernet v.1/v.2 electrical specifications. Contact your product support service for more information.
NAME
gpib – driver for National Instruments VME IEEE-488 controller

DESCRIPTION
The gpib driver provides access to instruments connected to the IEEE-488 instrument bus, also known as the GPIB, via the National Instruments GPIB-1014 VME controller. Up to 16 devices may be connected to a single controller card, using standard GPIB cabling.

The file `/usr/include/sys/ugpib.h` contains the ioctl and structure definitions needed for user programs to use the board. Both a read/write and an ioctl interface are provided. A library `libgpin.a` contains useful functions for controlling GPIB devices.

FILES
/dev/dev1 - /dev/dev16
/dev/gpib0

SEE ALSO
scanner(1)
The following manuals are available from National Instruments in Austin, Texas:

and
National Instruments GPIB-1014 User Manual, Part #320030-01

April 1990 - 1 - Version 5.0
NAME
gse — Silicon Graphics 5080 workstation interface card

DESCRIPTION
The special file /dev/gse provides access to the Silicon Graphics 5080 workstation interface card which connects a Silicon Graphics 4D system to a Channel Controller attached to an IBM mainframe. The connection of the interface card can be coax, V.35 or T1.

This special file is used by the IBM 5080 emulator (em5080(1)). The following control functions are provided by the driver to the emulator via ioctl(2).

```c
#include <sys/gseio.h>
struct gse_io gseio;

ioctl(fd,GIO_INIT,0); /* reset the controller */
ioctl(fd,GIO_SETIV,0); /* load the intr vector */
ioctl(fd,GIO_WAIT,timeout); /* wait for controller intr */
ioctl(fd,GIO_SIGNAL,signal); /* send signal on controller intr */
ioctl(fd,GIO_READ,&gseio); /* cp from controller to user buf */
ioctl(fd,GIO_WRITE,&gseio); /* cp from user buf to controller */
```

```c
struct gse_io {
    int     v_addr;    /* user virtual address */
    int     d_addr;    /* device real address */
    int     byte_ct;   /* byte count */
};
```

Access to the board from the application level emulator is also done through memory mapped addressing directly. The standard mmap(2) function is provided by the driver to set that up.

The standard select(2) function is also supported by the driver to facilitate the operation on the controller event from the emulator.

FILES
/dev/gse

SEE ALSO
cleanup(1), em3270(1), em5080(1), gateway(1), ld5080(1), sh5080(1)
NAME
hl – hardware spinlocks driver

DESCRIPTION
The hl driver manages the allocation and mapping of test-and-set hardware that is used as the basis of user spinlocks and semaphores.

Various driver routines and ioctl functions are available including those to allocate a shared lock group, allocate actual locks, and map the lock hardware into the user’s virtual address space. These driver routines are called from the various parallel processing library routines in libmpc.a.

Tset-and-set hardware is available only on the 4D GTX models.

CAVEATS
The hl driver is intended to be used only by the libmpc.a library. Client programs should communicate with libmpc.a for spinlocks and semaphores services, not with hl directly.

FILES
/dev/hl/hlock*

SEE ALSO
mmap(2), usinit(3P), usnewlock(3P), usnewsema(3P).
NAME
hy - HyperNet interface

DESCRIPTION
The hy interface provides a character and a network device interface to a
Network Systems Corporation A400 HYPERchannel(TM) Adaptor.

The network to which the interface is attached is specified with an SIOCSI-
FADDR ioctl. The routing information that maps internet addresses into
HYPERchannel addresses is supplied to the driver by the hyroute program.
The driver requires that both the routing tables and the interface address be
set before any packets can be sent through the network device interface.

Outgoing packets are stamped with the host's address for both the network
and character interface to prevent fraud.

SEE ALSO
inet(7F), hyroute(1M).

FILES
/usr/etc/hyroutef
/usr/etc/hyroute.conf
/etc/init.d/network
/etc/config/hypernet.on
/usr/etc/hydiag

NOTE
HYPERchannel is a registered trademark of Network Systems Corp.
NAME
icmp – Internet Control Message Protocol

SYNOPSIS
#include <sys/socket.h>
#include <netinet/in.h>

s = socket(AF_INET, SOCK_RAW, proto);

DESCRIPTION
ICMP is the error and control message protocol used by IP and the Internet
protocol family. It may be accessed through a "raw socket" for network
monitoring and diagnostic functions. The proto parameter to the socket call
to create an ICMP socket is obtained from getprotobyname(3N). ICMP
sockets are connectionless, and are normally used with the sendto and
recvfrom calls, though the connect(2) call may also be used to fix the desti-
nation for future packets (in which case the read(2) or recv(2) and write(2)
or send(2) system calls may be used).

Outgoing packets automatically have an IP header prepended to them
(based on the destination address). Incoming packets are received with the
IP header and options intact.

DIAGNOSTICS
A socket operation may fail with one of the following errors returned:

[EISCONN] when trying to establish a connection on a socket which
already has one, or when trying to send a datagram with
the destination address specified and the socket is already
connected;

[ENOTCONN] when trying to send a datagram, but no destination
address is specified, and the socket hasn’t been con-
nected;

[ENOBUFFS] when the system runs out of memory for an internal data
structure;

[EADDRNOTAVAIL] when an attempt is made to create a socket with a net-
work address for which no network interface exists.

SEE ALSO
send(2), recv(2), intro(3), inet(7F), ip(7P)
NAME

ik – Ikon 10088 hardcopy interface controller

SYNOPSIS

/dev/cent
/dev/vp0

DESCRIPTION

This is the driver for the Ikon 10088 (and 10088A) hardcopy interface controller. The controller can support any one of three possible interfaces: Versatec TTL, Versatec differential, or (Tektronix-compatible) Centronics. The Ikon controller board must be jumpered, switched, connected, and terminated differently depending on which interface is being used.

If the board is configured for the Centronics interface, bytes written to /dev/cent are output to the hardcopy port. A printer reset (input prime) is issued when the device file is opened.

If the board is configured for either Versatec interface, bytes written to /dev/vp0 are output to the hardcopy port. A printer reset is issued when the device file is opened. In addition, the following Versatec control calls are provided via ioctl(2). The relevant definitions are contained in the include file <sys/vcmd.h>.

ioctl(fd,VSETSTATE,int_3_ptr);

changes the Versatec device state according to the contents of the 3 integers pointed at by int_3_ptr. The first integer specifies a Versatec command:

VPRINT commands the Versatec device to enter print mode;
VPLLOT commands it to enter plot mode;
VPRINTPLOT commands it to enter simultaneous print/plot mode;
VLFL commands it to output a linefeed;
VFF commands it to output a formfeed;
VREOT drives the remote EOT signal.

The second integer specifies a timeout delay (in seconds) for subsequent Versatec operations, including output operations. A specified delay of 0 is ignored. The default delay is 300 seconds. This may be insufficient for large and / or slow devices. The third integer is ignored.

ioctl(fd,VGETSTATE,int_3_ptr);

passes back Versatec device state information in the 3 integers pointed at by int_3_ptr. The first integer contains:

VPRINT if the Versatec device is in print mode;
VPLLOT if it is in plot mode;
VPRINTPLOT if it is in simultaneous print/plot mode.
The second integer contains the current timeout delay (in seconds) (see VSETSTATE above). The third integer is not used.

It is normally unnecessary to use `/dev/cent` or `/dev/vp0` directly; the spooling software (see `lp(1)`) provides an adequate interface.

**FILES**

`/dev/cent`
`/dev/vp0`

**SEE ALSO**

`image(4), ipaste(1), lp(1), snap(1)`
NAME

imon – inode monitor device

SYNOPSIS

#include <sys/imon.h>
/dev/imon

DESCRIPTION

The inode monitor driver is a pseudo device driver which enables a user
level program to monitor filesystem activity on a file by file basis. The
application program expresses interest in specific files by means of an ioctl
request that specifies the pathname of the file and an indication of what
types of events are to be monitored. As various actions take place on a file
in which interest has been expressed, imon posts events through a queue
that may be read via the read system call.

Events

Calls to read return an integral number of imon queue elements. Each
queue element has the structure given below.

typedef struct {
    ino_t     qe_inode;  /* inode number of file */
    dev_t     qe_dev;    /* device of file */
    intmask_t qe_what;   /* what events occurred */
} qelem_t;

The qe_inode is a key that uniquely describes every file within a filesystem
and matches the st_ino field of the file’s stat structure (see stat(4)). The
qe_dev field similarly matches the st_dev field of the file’s stat structure.
These two fields together uniquely describe a file in the system. The third
field, qe_what, contains a bit-mask describing what event or events took
place on that file. The possible events are:

IMON_CONTENT

The contents or size of the file have changed. This is typically caused by a
write(2) call made by some process.

IMON_ATTRIBUTE

The mode or ownership have changed on the file. This is typically caused by
a chown(2) or chmod(2) system call.

IMON_DELETE

The last link to the file has gone away. When this event is sent, all interest in
the file is implicitly revoked. Note that if a process has the file open when
it is removed from the directory structure, this event will not be generated
until the file is closed.

**IMON_EXEC**

The file represents an executable command and a process has started executing that command. If multiple instances of the same command are subsequently started, an event is not generated. Therefore, the **IMON_EXEC** event only means that at least one process is executing from that file.

**IMON_EXIT**

The last process executing the file has exited.

**IMON_OVER**

The **imon** event queue has overflowed. When this occurs, the client process must re-express interest in each file to determine its true state.

Controls

The following structure is used by the **IMONIOC_EXPRESS** and **IMONIOC_REVOKE** controls described below.

```c
typedef struct {
    char * in_fname; /* pathname */
    struct stat * in_sb; /* optional status return buffer */
    intmask_t in_what; /* what types of events to send */
} interest_t;
```

**IMONIOC_EXPRESS**

Express interest in the file whose name is given in **in_fname**. If **in_fname** represents a symbolic link, the action takes place on the link itself, not the file to which it points. Only events in the bit-mask **in_what** will be generated. The **in_sb** field optionally points to a **stat(4)** buffer that will be filled in with the current state of the file. This allows the **imon** client to atomicly express interest and get the current state of a file. Multiple calls may be made on the same file and have a cumulative effect.
IMONIOC_REVOKE

Revoke interest in the file whose name is given by in_fname. The in_what field is used to determine which interests will be revoked. As with IMONIOC_EXPRESS, multiple calls may be made on the same file and have a cumulative effect.

IMONIOC_QTEST

Returns the number of events waiting to be read in the event queue.

BUGS

There is no way to accurately monitor the execution of a shell script with imon. Files in an NFS mounted filesystem will only generate events for things that happen as a result of local activity; changes made remotely on the NFS server will not be seen through imon.

CAVEATS

The imon driver is intended to be used only by the file access monitoring daemon (fam) and the interface is likely to change in future releases. Client programs should communicate with fam for monitoring services, not with imon directly.
NAME

inet – Internet protocol family

SYNOPSIS

#include <sys/types.h>
#include <netinet/in.h>

DESCRIPTION

The Internet protocol family is a collection of protocols layered atop the
Internet Protocol (IP) transport layer, and utilizing the Internet address for-
mat. The Internet family provides protocol support for the
SOCK_STREAM, SOCK_DGRAM, and SOCK_RAW socket types; the
SOCK_RAW interface provides access to the IP protocol.

ADDRESSING

Internet addresses are four byte quantities, stored in network standard for-
mat. The include file <netinet/in.h> defines this address as a discriminated
union.

Sockets bound to the Internet protocol family utilize the following address-
ing structure:

struct sockaddr_in {
    short sin_family;
    u_short sin_port;
    struct in_addr sin_addr;
    char sin_zero[8];
};

Sockets may be created with the local address INADDR_ANY to effect
"wildcard" matching on incoming messages. The address in a connect(2)
or sendto(2) call may be given as INADDR_ANY to mean "this host." The
distinguished address INADDR_BROADCAST is allowed as a short-
hand for the broadcast address on the primary network if the first network
configured supports broadcast.

PROTOCOLS

The Internet protocol family is comprised of the IP transport protocol, Internet
Control Message Protocol (ICMP), Internet Group Management Proto-
col (IGMP), Transmission Control Protocol (TCP), and User Datagram Pro-
tocol (UDP). TCP is used to support the SOCK_STREAM abstraction
while UDP is used to support the SOCK_DGRAM abstraction. A raw
interface to IP is available by creating an Internet socket of type
SOCK_RAW. The ICMP message protocol is accessible from a raw
socket.
The 32-bit Internet address contains both network and host parts. It is frequency-encoded; the most-significant bit is clear in Class A addresses, in which the high-order 8 bits are the network number. Class B addresses use the high-order 16 bits as the network field, and Class C addresses have a 24-bit network part. Sites with a cluster of local networks and a connection to the DARPA Internet may chose to use a single network number for the cluster; this is done by using subnet addressing. The local (host) portion of the address is further subdivided into subnet and host parts. Within a subnet, each subnet appears to be an individual network; externally, the entire cluster appears to be a single, uniform network requiring only a single routing entry. Subnet addressing is enabled and examined by the following *ioctl*(2) commands on a datagram socket in the Internet domain; they have the same form as the SIOCIFADDR command (see *netintro*(7)).

**SIOCSIFNETMASK** Set interface network mask. The network mask defines the network part of the address; if it contains more of the address than the address type would indicate, then subnets are in use.

**SIOCGIFNETMASK** Get interface network mask.

**SEE ALSO**


Network Communications Guide.

**CAVEAT**

The Internet protocol support is subject to change as the Internet protocols develop. Users should not depend on details of the current implementation, but rather the services exported.
NAME
ip – Internet Protocol

SYNOPSIS
#include <sys/socket.h>
#include <netinet/in.h>

s = socket(AF_INET, SOCK_RAW, proto);

DESCRIPTION
IP is the transport layer protocol used by the Internet protocol family. Options may be set at the IP level when using higher-level protocols that are based on IP (such as TCP and UDP). It may also be accessed through a “raw socket” when developing new protocols, or special purpose applications.

A single generic option is supported at the IP level, IP_OPTIONS, that may be used to provide IP options to be transmitted in the IP header of each outgoing packet. Options are examined with getsockopt(2). The format of IP options to be sent is that specified by the IP protocol specification, with one exception: the list of addresses for Source Route options must include the first-hop gateway at the beginning of the list of gateways. The first-hop gateway address will be extracted from the option list and the size adjusted accordingly before use. IP options may be used with any socket type in the Internet family.

Raw IP sockets are connectionless, and are normally used with the sendto and recvfrom calls, though the connect(2) call may also be used to fix the destination for future packets (in which case the read(2) or recv(2) and write(2) or send(2) system calls may be used).

If proto is 0, the default protocol IPPROTO_RAW is used for outgoing packets, and only incoming packets destined for that protocol are received. If proto is non-zero, that protocol number will be used on outgoing packets and to filter incoming packets.

Outgoing packets automatically have an IP header prepended to them (based on the destination address and the protocol number the socket is created with). Incoming packets are received with IP header and options intact.

DIAGNOSTICS
A socket operation may fail with one of the following errors returned:

[EISCONN] when trying to establish a connection on a socket which already has one, or when trying to send a datagram with the destination address specified and the socket is already connected;
[ENOTCONN] when trying to send a datagram, but no destination address is specified, and the socket hasn’t been connected;

[ENOBUFS] when the system runs out of memory for an internal data structure;

[EADDRNOTAVAIL] when an attempt is made to create a socket with a network address for which no network interface exists.

The following errors specific to IP may occur when setting or getting IP options:

[EINVAL] An unknown socket option name was given.

[EINVAL] The IP option field was improperly formed; an option field was shorter than the minimum value or longer than the option buffer provided.

SEE ALSO
getsockopt(2), send(2), recv(2), intro(3), icmp(7P), inet(7F)
NAME

ipi, xylipi - Xylogics IPI disk controllers and driver.

SYNOPSIS

/dev/dsk/ipi*
/dev/rdsk/ipi*

DESCRIPTION

The IRIS-4D system supports the Xylogics SV-7800 IPI disk controller. A system may contain a maximum of 2 IPI controllers. Each controller has two channels, each of which may have 8 disk drives. Thus, each controller may have up to 16 IPI disk drives connected to it.

The special files are named according to the convention discussed in intro(7M):

/dev/{r}dsk/ipi<controller->d<drive->{s<partition->vh|vol}

Controller numbers run from 0 to 1 for the two possible controllers in a system. Drive numbers run from 0 to 7 on channel 0 (the lower connector) and from 8 to 15 on channel 1 (the upper connector). The drive ID, or drive number, should be relative to the first drive on a channel. Thus partition 7 on drive 3 on channel 1 on controller 1 would be referred to as ipi1d11s7.

The kernel driver for the IPI controller is referred to as xylipi.

IOCTL FUNCTIONS

As well as normal read and write operations, the driver supports a number of special io control (ioctl) operations when opened via the character special file in /dev/rdsk. These may be invoked from a user program by the ioctl system call, see ioctl(2). Command values for these are defined in the system include file <sys/dkio.h>

These ioctl operations are intended for the use of special-purpose disk utilities. Many of them can have drastic or even fatal effects on disk operation if misused; they should be invoked only by the knowledgeable and with extreme caution!

A list of the ioctl commands supported by the xylipi driver is given below.

DIOCGETVH

reads the disk volume header from the driver into a buffer in the calling program. The arg in the ioctl call is expected to point to a buffer of size at least 512 bytes. The structure of the volume header is defined in the include file <sys/dvh.h>.
DIOCHANDSHAKE
reads identifying data from the controller firmware into a buffer in
the calling program. The arg of the ioctl call should point to a
buffer at least 16 bytes long. The identifying data is a 'struct
CONTROLLER_TABLE', defined in <sys/x1.h>.

DIOCSENSE
causes the current controller parameters to be printed on the sys-
tem console. These are the fields of a 'struct
CONTROLLER_TABLE', defined in <sys/x1.h>.

The remaining commands should be used with extreme caution,
since misuse can render a disk at least temporarily unreadable, or
at worst destroy data on the disk.

DIOCSETVH
transfers a new volume header into the driver, and causes the disk
drive to be reconfigured in accordance with the parameters in the
new volume header. The arg in the ioctl call is expected to point
to a buffer containing a valid volume header with parameters
appropriate for the drive.

DIOCFMTMAP
formats a track on the disk, or maps a track to a replacement track,
according to information passed in a buffer pointed to by the arg of
the ioctl call. The arg should point to a 'struct fmt_map_info',
which is defined in the include file <sys/dvh.h>. Note that
although this contains definitions for a number of other possible
actions, only formatting a track and mapping a track are currently
supported. The track(s) to be acted on must be specified in the
form of cylinder/head numbers, and these are absolute numbers
starting from the beginning of the disk (that is to say, NOT from
start of the currently open partition).

FILES
/dev/dsk/ipi*
/dev/rdsk/ipi*
/usr/include/sys/dkio.h /usr/include/sys/dvh.h /usr/include/sys/x1.h

BUGS
For historical reasons, the names of the ioctl's are not always logical, and
their arguments are an irregular mess.
NAME
ips, dkip – Interphase disk controllers and driver.

SYNOPSIS
/dev/dsk/ips*
/dev/rdsk/ips*

DESCRIPTION
The IRIS-4D system supports two types of Interphase disk controller: the 3201 and 4201 ESDI controllers. The 3201 supports up to two drives per controller, the 4201 supports up to 4 drives per controller. A system may contain a maximum of 2 Interphase controllers; types may be mixed.

Note that the 3201 is regarded as obsolescent and is not recommended for new systems.

The special files are named according to the convention discussed in intro(7M):

/dev/{r}dsk/ips<controller-#>d<drive-#>{s<partition-#>|vh|vol}

Controller numbers run from 0 to 1 for the two possible controllers in a system. Drive numbers run from 0 to 1 for the 3201, and from 0 to 3 for the 4201.

There is one common kernel driver, referred to as dkip, for all the Interphase controllers.

IOCTL FUNCTIONS
As well as normal read and write operations, the driver supports a number of special io control (ioctl) operations when opened via the character special file in /dev/rdsk. These may be invoked from a user program by the ioctl system call, see ioctl(2). Command values for these are defined in the system include file <sys/dkio.h>.

These ioctl operations are intended for the use of special-purpose disk utilities. Many of them can have drastic or even fatal effects on disk operation if misused; they should be invoked only by the knowledgeable and with extreme caution!

A list of the ioctl commands supported by the dkip driver is given below.

DIOCGETVH
reads the disk volume header from the driver into a buffer in the calling program. The arg in the ioctl call is expected to point to a buffer of size at least 512 bytes. The structure of the volume

April 1990 - 1 - Version 5.0
header is defined in the include file <sys/dvh.h>.

DIOCHANDSHAKE
reads identifying data from the controller firmware into a buffer in the calling program. The arg of the ioctl call should point to a buffer at least 16 bytes long. The structure of the identifying data is defined in the include file <sys/dkipreg.h>.

DIOCRAWREAD
This is something of a misnomer; what it actually does is to invoke a controller command for reading the flawmap information placed on the disk by the original manufacturer in accordance with the ESDI specification. The amount of data returned to the calling program is always 512 bytes and the program must provide a buffer of this size to receive it. The operation is controlled by a structure which must be pointed to by the arg of the ioctl call; this is a 'struct rawread_info' which is defined in <sys/dvh.h>. Flawmaps are always at the start of tracks, however for historical reasons the identifying argument is an absolute blocknumber rather than a cylinder/head specification. Thus it is necessary to know the geometry of the disk in order to construct the arguments. For the interpretation of the returned data, see ESDI interface specifications.

DIOCCDCRAW
is a variant of the previous command for use on CDC disks where the way in which the defect information is recorded differs slightly from other manufacturers.

DIOCSENSE
causes the Unit Initialization Block used by the controller for the current drive to be printed on the system console. This contains disk geometry and format parameters, the structure members are defined in <sys/dkipreg.h>.

The remaining commands should be used with extreme caution, since misuse can render a disk at least temporarily unreadable, or at worst destroy data on the disk.

DIOCSETVH
transfers a new volume header into the driver, and causes the disk drive to be reconfigured in accordance with the parameters in the new volume header. The arg in the ioctl call is expected to point to a buffer containing a valid volume header with parameters appropriate for the drive.
DIOCFMTMAP

formats a track on the disk, or maps a track to a replacement track, 
according to information passed in a buffer pointed to by the arg of 
the ioctl call. The arg should point to a 'struct fmt_map_info', 
which is defined in the include file <sys/dvh.h>. Note that 
although this contains definitions for a number of other possible 
actions, only formatting a track and mapping a track are currently 
supported. The track(s) to be acted on must be specified in the 
form of cylinder/head numbers, and these are absolute numbers 
starting from the beginning of the disk (that is to say, NOT from 
start of the currently open partition).

FILES

/dev/dsk/ips*
/dev/rdsk/ips*
/usr/include/sys/dkio.h /usr/include/sys/dvh.h /usr/include/sys/dkipreg.h

BUGS

For historical reasons, the names of the ioctls are not always logical, and 
their arguments are an irregular mess. 
DIOCSENSE should probably have an option to return data to the caller 
rather than always printing on the system console.
NAME

keyboard – keyboard specifications

DESCRIPTION

OVERVIEW

The keyboard is an up-down encoded 101 key keyboard.

The keyboard connects to the main electronics cabinet through a shielded partially coiled cord, and is detachable at the system cabinet only. The optical mouse plugs into either side of the keyboard. Ports are provided on both sides of the enclosure to allow access to left-handed and right-handed mouse connectors. The keyboard cord contains low voltage direct current power feeds and two serial links; one for the mouse and one for the keyboard. The keyboard serial link is bidirectional, allowing for control of indicator lights and other keyboard functions. Each time a key is pressed or released, a code is sent via the keyboard serial link. Every key has a different upcode and downcode. All keys function the same way, allowing the system software to use keys in any manner. Auto-repeat is the only function that treats keys differently. When auto-repeat is enabled, a subset of the keys will repeat when held down. Multiple key presses/releases result in all key transitions being reported.

ELECTRICAL INTERFACE

The keyboard serial I/O interface uses RS423 levels and communicates asynchronously to the system at 600 baud. The format used is one start bit followed by eight data bits, an odd parity bit and one stop bit, with one byte sent per key up or down transition. The idle state and true data bits for the interface are Mark level or -5V, whereas false data bits and the start bit are spaces or +5V. The pin assignments for the mouse port connector are shown in the following table:

<table>
<thead>
<tr>
<th>MOUSE PORTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIN</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>9</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>3</td>
</tr>
</tbody>
</table>
The pin assignments for the keyboard connector are shown in the following table:

<table>
<thead>
<tr>
<th>KYBD CABLE PINOUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signal</td>
</tr>
<tr>
<td>GND</td>
</tr>
<tr>
<td>KTXD</td>
</tr>
<tr>
<td>KRCD</td>
</tr>
<tr>
<td>MTXD</td>
</tr>
<tr>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
</tr>
<tr>
<td>+12Vdc</td>
</tr>
<tr>
<td>+12Vdc</td>
</tr>
<tr>
<td>GND</td>
</tr>
<tr>
<td>GND</td>
</tr>
<tr>
<td>-12Vdc</td>
</tr>
<tr>
<td>+12Vdc</td>
</tr>
</tbody>
</table>

SOFTWARE INTERFACE

The interface between the keyboard and the system is 600 baud asynchronous. The format used is one start bit followed by eight data bits, an odd parity bit and one stop bit, with one byte sent per key up or down transition. The MSB of the byte is a "0" for a downstroke and a "1" for an upstroke. Control bytes are sent to the keyboard with the same speed and format. The system software does all the processing needed to support functions such as capitalization, control characters, and numeric lock. Auto-repeat for a specified set of characters can be turned on or off by the system software by sending a control byte to the keyboard. When auto-repeat is enabled a pressed key will begin auto-repeating after 0.65 seconds and repeat 28 times per second. The keyboard initializes upon power-up. The configuration request control byte causes the keyboard to send a two-byte sequence to the system. The second byte contains the eight-bit value set on a DIP switch in the keyboard. All keyboard lights are controlled by the system software by sending control bytes to the keyboard to turn them on or off. Control bytes are also used for long and short beep control and key click disable. When key click is enabled, the keys click when they are pressed. The long beep duration is 1 second and the short beep duration is 0.2 second. There are four general-purpose keyboard lights labeled L1 through L4 and three lights labeled NUM LOCK, CAPS LOCK, and SCROLL LOCK. The required keycode mappings and control byte formats are shown in the following tables. Note that the legend names prefixed by two asterisks are reserved, and not implemented on the keyboard. Legend
names prefixed by two exclamation marks do NOT have the auto-repeat enable capability. Legend names prefixed by two dollar signs do NOT have the key click enable capability.
<table>
<thead>
<tr>
<th>Legend</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>AKEY</td>
<td>10</td>
</tr>
<tr>
<td>BKEY</td>
<td>35</td>
</tr>
<tr>
<td>CKEY</td>
<td>27</td>
</tr>
<tr>
<td>DKEY</td>
<td>17</td>
</tr>
<tr>
<td>EKEY</td>
<td>16</td>
</tr>
<tr>
<td>FKEY</td>
<td>18</td>
</tr>
<tr>
<td>GKEY</td>
<td>25</td>
</tr>
<tr>
<td>HKEY</td>
<td>26</td>
</tr>
<tr>
<td>IKEY</td>
<td>39</td>
</tr>
<tr>
<td>JKEY</td>
<td>33</td>
</tr>
<tr>
<td>KKEY</td>
<td>34</td>
</tr>
<tr>
<td>LKEY</td>
<td>41</td>
</tr>
<tr>
<td>MKEY</td>
<td>43</td>
</tr>
<tr>
<td>NKEY</td>
<td>36</td>
</tr>
<tr>
<td>OKEY</td>
<td>40</td>
</tr>
<tr>
<td>PKEY</td>
<td>47</td>
</tr>
<tr>
<td>QKEY</td>
<td>9</td>
</tr>
<tr>
<td>RKEY</td>
<td>23</td>
</tr>
<tr>
<td>SKEY</td>
<td>11</td>
</tr>
<tr>
<td>TKEY</td>
<td>24</td>
</tr>
<tr>
<td>UKEY</td>
<td>32</td>
</tr>
<tr>
<td>VKEY</td>
<td>28</td>
</tr>
<tr>
<td>WKEY</td>
<td>15</td>
</tr>
<tr>
<td>XKEY</td>
<td>20</td>
</tr>
<tr>
<td>YKEY</td>
<td>31</td>
</tr>
<tr>
<td>ZKEY</td>
<td>19</td>
</tr>
<tr>
<td>ZEROKEY</td>
<td>45</td>
</tr>
<tr>
<td>ONEKEY</td>
<td>7</td>
</tr>
<tr>
<td>TWOKEY</td>
<td>13</td>
</tr>
<tr>
<td>THREEKEY</td>
<td>14</td>
</tr>
<tr>
<td>FOURKEY</td>
<td>21</td>
</tr>
<tr>
<td>FIVEKEY</td>
<td>22</td>
</tr>
<tr>
<td>SIXKEY</td>
<td>29</td>
</tr>
<tr>
<td>SEVENKEY</td>
<td>30</td>
</tr>
<tr>
<td>EIGHTKEY</td>
<td>37</td>
</tr>
<tr>
<td>NINEKEY</td>
<td>38</td>
</tr>
<tr>
<td>Legend</td>
<td>Code</td>
</tr>
<tr>
<td>-------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td><strong>!!BREAKKEY</strong></td>
<td>0</td>
</tr>
<tr>
<td><strong>!!SETUPKEY</strong></td>
<td>1</td>
</tr>
<tr>
<td>$!!LEFTCTRL</td>
<td>2</td>
</tr>
<tr>
<td>$!!CAPSLOCKKEY</td>
<td>3</td>
</tr>
<tr>
<td>$!!RIGHTSHIFTKEY</td>
<td>4</td>
</tr>
<tr>
<td>$!!LEFTSHIFTKEY</td>
<td>5</td>
</tr>
<tr>
<td><strong>!!NOSCRLKEY</strong></td>
<td>12</td>
</tr>
<tr>
<td>!!ESCKEY</td>
<td>6</td>
</tr>
<tr>
<td>!!TABKEY</td>
<td>8</td>
</tr>
<tr>
<td>RETURN.ENTER</td>
<td>50</td>
</tr>
<tr>
<td>SPACEKEY</td>
<td>82</td>
</tr>
<tr>
<td><strong>LINEFEEDKEY</strong></td>
<td>59</td>
</tr>
<tr>
<td>BACKSPACEKEY</td>
<td>60</td>
</tr>
<tr>
<td>DELKEY</td>
<td>61</td>
</tr>
<tr>
<td>SEMICOLONKEY</td>
<td>42</td>
</tr>
<tr>
<td>PERIODKEY</td>
<td>51</td>
</tr>
<tr>
<td>COMMAKEY</td>
<td>44</td>
</tr>
<tr>
<td>QUOTEKEY&quot;</td>
<td>49</td>
</tr>
<tr>
<td>ACCENTGRAVEKEY&quot;</td>
<td>54</td>
</tr>
<tr>
<td>MINUSKEY</td>
<td>46</td>
</tr>
<tr>
<td>VIRGULEKEY?</td>
<td>52</td>
</tr>
<tr>
<td>BACKSLASHKEY</td>
<td>56</td>
</tr>
<tr>
<td>EQUALKEY</td>
<td>53</td>
</tr>
<tr>
<td>LEFTBRACKETKEY</td>
<td>48</td>
</tr>
<tr>
<td>RIGHTBRACKETKEY</td>
<td>55</td>
</tr>
<tr>
<td>LEFTARROWKEY</td>
<td>72</td>
</tr>
<tr>
<td>DOWNARROWKEY</td>
<td>73</td>
</tr>
<tr>
<td>RIGHTARROWKEY</td>
<td>79</td>
</tr>
<tr>
<td>UPARROWKEY</td>
<td>80</td>
</tr>
<tr>
<td>PAD0</td>
<td>58</td>
</tr>
<tr>
<td>PAD1</td>
<td>57</td>
</tr>
<tr>
<td>PAD2</td>
<td>63</td>
</tr>
<tr>
<td>PAD3</td>
<td>64</td>
</tr>
<tr>
<td>PAD4</td>
<td>62</td>
</tr>
<tr>
<td>PAD5</td>
<td>68</td>
</tr>
<tr>
<td>PAD6</td>
<td>69</td>
</tr>
<tr>
<td>Legend</td>
<td>Code</td>
</tr>
<tr>
<td>---------------------</td>
<td>------</td>
</tr>
<tr>
<td>PAD7</td>
<td>66</td>
</tr>
<tr>
<td>PAD8</td>
<td>67</td>
</tr>
<tr>
<td>PAD9</td>
<td>74</td>
</tr>
<tr>
<td>**PADPF1</td>
<td>71</td>
</tr>
<tr>
<td>**PADPF2</td>
<td>70</td>
</tr>
<tr>
<td>**PADPF3</td>
<td>78</td>
</tr>
<tr>
<td>**PADPF4</td>
<td>77</td>
</tr>
<tr>
<td>PADPERIOD</td>
<td>65</td>
</tr>
<tr>
<td>PADMINUS</td>
<td>75</td>
</tr>
<tr>
<td>**PADCOMMA</td>
<td>76</td>
</tr>
<tr>
<td>!!PADENTER</td>
<td>81</td>
</tr>
<tr>
<td>$$!!LEFTALT</td>
<td>83</td>
</tr>
<tr>
<td>$$!!RIGHTALT</td>
<td>84</td>
</tr>
<tr>
<td>$$!!RIGHTCTRL</td>
<td>85</td>
</tr>
<tr>
<td>F1</td>
<td>86</td>
</tr>
<tr>
<td>F2</td>
<td>87</td>
</tr>
<tr>
<td>F3</td>
<td>88</td>
</tr>
<tr>
<td>F4</td>
<td>89</td>
</tr>
<tr>
<td>F5</td>
<td>90</td>
</tr>
<tr>
<td>F6</td>
<td>91</td>
</tr>
<tr>
<td>F7</td>
<td>92</td>
</tr>
<tr>
<td>F8</td>
<td>93</td>
</tr>
<tr>
<td>F9</td>
<td>94</td>
</tr>
<tr>
<td>F10</td>
<td>95</td>
</tr>
<tr>
<td>F11</td>
<td>96</td>
</tr>
<tr>
<td>F12</td>
<td>97</td>
</tr>
<tr>
<td>$$!!PRINT.SCREEN</td>
<td>98</td>
</tr>
<tr>
<td>$$!!SCROLL.LOCK</td>
<td>99</td>
</tr>
<tr>
<td>!!PAUSE</td>
<td>100</td>
</tr>
<tr>
<td>!!INSERT</td>
<td>101</td>
</tr>
<tr>
<td>!!HOME</td>
<td>102</td>
</tr>
<tr>
<td>!!PAGEUP</td>
<td>103</td>
</tr>
<tr>
<td>!!END</td>
<td>104</td>
</tr>
<tr>
<td>!!PAGEDOWN</td>
<td>105</td>
</tr>
<tr>
<td>$$!!NUM.LOCK</td>
<td>106</td>
</tr>
<tr>
<td>PAD.BKSLASH/</td>
<td>107</td>
</tr>
<tr>
<td>Legend</td>
<td>Code</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-------</td>
</tr>
<tr>
<td>PAD.ASTER*</td>
<td>108</td>
</tr>
<tr>
<td>PAD.PLUS+</td>
<td>109</td>
</tr>
<tr>
<td>config byte (1st of 2 bytes)</td>
<td>110</td>
</tr>
<tr>
<td>config byte (2nd of 2 bytes)</td>
<td>DIP SW</td>
</tr>
<tr>
<td>GERLessThan</td>
<td>111</td>
</tr>
<tr>
<td>spare1</td>
<td>112</td>
</tr>
<tr>
<td>spare2</td>
<td>113</td>
</tr>
<tr>
<td>spare3</td>
<td>114</td>
</tr>
<tr>
<td>spare4</td>
<td>115</td>
</tr>
<tr>
<td>spare6</td>
<td>117</td>
</tr>
<tr>
<td>spare7</td>
<td>118</td>
</tr>
<tr>
<td>spare8</td>
<td>119</td>
</tr>
<tr>
<td>spare9</td>
<td>120</td>
</tr>
<tr>
<td>spare10</td>
<td>121</td>
</tr>
<tr>
<td>Code</td>
<td>Legend</td>
</tr>
<tr>
<td>------</td>
<td>----------------------------</td>
</tr>
<tr>
<td>0</td>
<td>**BREAKKEY</td>
</tr>
<tr>
<td>1</td>
<td>**!!SETUPKEY</td>
</tr>
<tr>
<td>2</td>
<td>$$!!LEFTCTRL</td>
</tr>
<tr>
<td>3</td>
<td>$$!!CAPSLOCKKEY</td>
</tr>
<tr>
<td>4</td>
<td>$$!!RIGHTSHIFTKEY</td>
</tr>
<tr>
<td>5</td>
<td>$$!!LEFTSHIFTKEY</td>
</tr>
<tr>
<td>6</td>
<td>!!ESCKEY</td>
</tr>
<tr>
<td>7</td>
<td>ONEKEY</td>
</tr>
<tr>
<td>8</td>
<td>!!TABKEY</td>
</tr>
<tr>
<td>9</td>
<td>QKEY</td>
</tr>
<tr>
<td>10</td>
<td>AKEY</td>
</tr>
<tr>
<td>11</td>
<td>SKEY</td>
</tr>
<tr>
<td>12</td>
<td>**!!NOSCRLKEY</td>
</tr>
<tr>
<td>13</td>
<td>TWOKEY</td>
</tr>
<tr>
<td>14</td>
<td>THREEKEY</td>
</tr>
<tr>
<td>15</td>
<td>WKKEY</td>
</tr>
<tr>
<td>16</td>
<td>EKEY</td>
</tr>
<tr>
<td>17</td>
<td>DKEY</td>
</tr>
<tr>
<td>18</td>
<td>FKEY</td>
</tr>
<tr>
<td>19</td>
<td>ZKEY</td>
</tr>
<tr>
<td>20</td>
<td>XKEY</td>
</tr>
<tr>
<td>21</td>
<td>FOURKEY</td>
</tr>
<tr>
<td>22</td>
<td>FIVEKEY</td>
</tr>
<tr>
<td>23</td>
<td>RKEY</td>
</tr>
<tr>
<td>24</td>
<td>TKEY</td>
</tr>
<tr>
<td>25</td>
<td>GKEY</td>
</tr>
<tr>
<td>26</td>
<td>HKEY</td>
</tr>
<tr>
<td>27</td>
<td>CKEY</td>
</tr>
<tr>
<td>28</td>
<td>VKEY</td>
</tr>
<tr>
<td>29</td>
<td>SIXKEY</td>
</tr>
<tr>
<td>30</td>
<td>SEVENKEY</td>
</tr>
<tr>
<td>31</td>
<td>YKEY</td>
</tr>
<tr>
<td>32</td>
<td>UKEY</td>
</tr>
<tr>
<td>33</td>
<td>JKEY</td>
</tr>
<tr>
<td>34</td>
<td>KKEY</td>
</tr>
<tr>
<td>35</td>
<td>BKEY</td>
</tr>
<tr>
<td>Code</td>
<td>Legend</td>
</tr>
<tr>
<td>------</td>
<td>----------------------------</td>
</tr>
<tr>
<td>36</td>
<td>NKEY</td>
</tr>
<tr>
<td>37</td>
<td>EIGHTKEY</td>
</tr>
<tr>
<td>38</td>
<td>NINEKEY</td>
</tr>
<tr>
<td>39</td>
<td>IKEY</td>
</tr>
<tr>
<td>40</td>
<td>OKEY</td>
</tr>
<tr>
<td>41</td>
<td>LKEY</td>
</tr>
<tr>
<td>42</td>
<td>SEMICOLONKEY</td>
</tr>
<tr>
<td>43</td>
<td>MKEY</td>
</tr>
<tr>
<td>44</td>
<td>COMMAKEY</td>
</tr>
<tr>
<td>45</td>
<td>ZEROKEY</td>
</tr>
<tr>
<td>46</td>
<td>MINUSKEY</td>
</tr>
<tr>
<td>47</td>
<td>PKEY</td>
</tr>
<tr>
<td>48</td>
<td>LEFTBRACKET</td>
</tr>
<tr>
<td>49</td>
<td>QUOTEKEY</td>
</tr>
<tr>
<td>50</td>
<td>RETURN, ENTER</td>
</tr>
<tr>
<td>51</td>
<td>PERIODKEY</td>
</tr>
<tr>
<td>52</td>
<td>VIRGULEKEY</td>
</tr>
<tr>
<td>53</td>
<td>EQUALKEY</td>
</tr>
<tr>
<td>54</td>
<td>ACCENTGRAVEKEY</td>
</tr>
<tr>
<td>55</td>
<td>RIGHTBRACKETKEY</td>
</tr>
<tr>
<td>56</td>
<td>BACKSLASHKEY</td>
</tr>
<tr>
<td>57</td>
<td>PADONEKEY</td>
</tr>
<tr>
<td>58</td>
<td>PADZEROKEY</td>
</tr>
<tr>
<td>59</td>
<td>**LINEFEEDKEY</td>
</tr>
<tr>
<td>60</td>
<td>BACKSPACEKEY</td>
</tr>
<tr>
<td>61</td>
<td>DELETETKEY</td>
</tr>
<tr>
<td>62</td>
<td>PADFOURKEY</td>
</tr>
<tr>
<td>63</td>
<td>PADTWOKEY</td>
</tr>
<tr>
<td>64</td>
<td>PADTHREEKEY</td>
</tr>
<tr>
<td>65</td>
<td>PADPERIODKEY</td>
</tr>
<tr>
<td>66</td>
<td>PADSEVENKEY</td>
</tr>
<tr>
<td>67</td>
<td>PADEIGHTKEY</td>
</tr>
<tr>
<td>68</td>
<td>PADFIVEKEY</td>
</tr>
<tr>
<td>69</td>
<td>PADSIXKEY</td>
</tr>
<tr>
<td>70</td>
<td>**PADPF2KEY</td>
</tr>
<tr>
<td>71</td>
<td>**PADPF1KEY</td>
</tr>
<tr>
<td>Code</td>
<td>Legend</td>
</tr>
<tr>
<td>------</td>
<td>-------------------------</td>
</tr>
<tr>
<td>72</td>
<td>LEFTARROWKEY</td>
</tr>
<tr>
<td>73</td>
<td>DOWNARROWKEY</td>
</tr>
<tr>
<td>74</td>
<td>PADNINEKEY</td>
</tr>
<tr>
<td>75</td>
<td>PADMINUSKEY</td>
</tr>
<tr>
<td>76</td>
<td>**PADCOMMAKEY</td>
</tr>
<tr>
<td>77</td>
<td>**PADPF4KEY</td>
</tr>
<tr>
<td>78</td>
<td>**PADPF3KEY</td>
</tr>
<tr>
<td>79</td>
<td>RIGHTARROWKEY</td>
</tr>
<tr>
<td>80</td>
<td>UPARROWKEY</td>
</tr>
<tr>
<td>81</td>
<td>!!PADENTERKEY</td>
</tr>
<tr>
<td>82</td>
<td>SPACEKEY</td>
</tr>
<tr>
<td>83</td>
<td>$$!!LEFTALT</td>
</tr>
<tr>
<td>84</td>
<td>$$!!RIGHTALT</td>
</tr>
<tr>
<td>85</td>
<td>$$!!RIGHTCTRL</td>
</tr>
<tr>
<td>86</td>
<td>F1</td>
</tr>
<tr>
<td>87</td>
<td>F2</td>
</tr>
<tr>
<td>88</td>
<td>F3</td>
</tr>
<tr>
<td>89</td>
<td>F4</td>
</tr>
<tr>
<td>90</td>
<td>F5</td>
</tr>
<tr>
<td>91</td>
<td>F6</td>
</tr>
<tr>
<td>92</td>
<td>F7</td>
</tr>
<tr>
<td>93</td>
<td>F8</td>
</tr>
<tr>
<td>94</td>
<td>F9</td>
</tr>
<tr>
<td>95</td>
<td>F10</td>
</tr>
<tr>
<td>96</td>
<td>F11</td>
</tr>
<tr>
<td>97</td>
<td>F12</td>
</tr>
<tr>
<td>98</td>
<td>!!PRINTSCREEN</td>
</tr>
<tr>
<td>99</td>
<td>$$!!SCROLLLOCK</td>
</tr>
<tr>
<td>100</td>
<td>!!PAUSE</td>
</tr>
<tr>
<td>101</td>
<td>!!INSERT</td>
</tr>
<tr>
<td>102</td>
<td>!!HOME</td>
</tr>
<tr>
<td>103</td>
<td>!!PAGEUP</td>
</tr>
<tr>
<td>104</td>
<td>!!END</td>
</tr>
<tr>
<td>105</td>
<td>!!PAGEDOWN</td>
</tr>
<tr>
<td>106</td>
<td>$$!!NUMLOCK</td>
</tr>
<tr>
<td>107</td>
<td>PAD.BKSLASH/</td>
</tr>
</tbody>
</table>
### KEYCODES IN DECIMAL VS LEGENDS

<table>
<thead>
<tr>
<th>Code</th>
<th>Legend</th>
</tr>
</thead>
<tbody>
<tr>
<td>108</td>
<td>PAD.ASTER*</td>
</tr>
<tr>
<td>109</td>
<td>PAD.PLUS+</td>
</tr>
<tr>
<td>110</td>
<td>config byte(1st of 2 bytes)</td>
</tr>
<tr>
<td>DIP SW</td>
<td>config byte(2nd of 2 bytes)</td>
</tr>
</tbody>
</table>

### CONTROL BYTES RECOGNIZED BY KEYBOARD

<table>
<thead>
<tr>
<th>BIT TRUE</th>
<th>DESCRIPTION</th>
<th>BIT 0 = 0</th>
<th>BIT 0 = 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>short beep</td>
<td></td>
<td>complement ds1 and ds2</td>
</tr>
<tr>
<td>2</td>
<td>long beep</td>
<td></td>
<td>ds3</td>
</tr>
<tr>
<td>3</td>
<td>click disable</td>
<td></td>
<td>ds4</td>
</tr>
<tr>
<td>4</td>
<td>return configuration byte</td>
<td></td>
<td>ds5</td>
</tr>
<tr>
<td>5</td>
<td>ds1</td>
<td></td>
<td>ds6</td>
</tr>
<tr>
<td>6</td>
<td>ds2</td>
<td></td>
<td>ds7</td>
</tr>
<tr>
<td>7</td>
<td>enable auto-repeat</td>
<td></td>
<td>not used</td>
</tr>
</tbody>
</table>

### DISPLAY LABELS

<table>
<thead>
<tr>
<th>DISPLAY DESIGNATION</th>
<th>LABEL</th>
</tr>
</thead>
<tbody>
<tr>
<td>ds1</td>
<td>NUM LOCK</td>
</tr>
<tr>
<td>ds2</td>
<td>CAPS LOCK</td>
</tr>
<tr>
<td>ds3</td>
<td>SCROLL LOCK</td>
</tr>
<tr>
<td>ds4</td>
<td>L1</td>
</tr>
<tr>
<td>ds5</td>
<td>L2</td>
</tr>
<tr>
<td>ds6</td>
<td>L3</td>
</tr>
<tr>
<td>ds7</td>
<td>L4</td>
</tr>
</tbody>
</table>
NAME
klog – kernel error logging interface

DESCRIPTION
Minor device 0 of the klog driver is the interface between a process and the kernel’s error logging mechanism. When this device is open, messages printed by the kernel, which normally appear only in the system console window, also are buffered by the klog driver. The data obtained by reading from this driver are the text of the kernel messages. The driver may be opened only for reading by a single process. Each read causes the requested number of bytes to be retrieved; the request is truncated if the read request is for more than the data currently available.

Normally, this device is opened and read by syslogd(1M), the system logging daemon.

This special device supports the following open(2) flags:

**O_NDELAY**
When set a read will not block if no data is present. If not set, a read will block until information becomes available.

Each ioctl(2) call that this special device supports has the form:

`ioctl(fd, cmd, arg)`

where the format and meaning of `arg` vary with the specified `cmd` as described below:

**FIONREAD**
Returns the number of bytes currently available to read to the area pointed to by `arg`, where `arg` is treated as a pointer to a variable of type `int`.

**FIONBIO**
Enables or disables non-blocking mode, according to the boolean value supplied as `arg`. Enabling this mode has the same affect as the **O_NDELAY** flag for open(2).

**FIOASYNC**
Enables or disables asynchronous mode, according to the boolean value supplied as `arg`. When asynchronous mode is enabled, the process group associated with the open of the special device receives a **SIGIO** signal when data is available to be read.

**TIOCSPGRP**
Set the process group for use with asynchronous I/O (FIOASYNC). For this command, `arg` should contain the process group id.
TIOCGPGRP
Returns the current process group to the area pointed to by arg, where arg is treated as a pointer to a variable of type int.

FILES
/dev/klog special file

SEE ALSO
open(2), ioctl(2), syslogd(1M)
NAME
lv – logical volume Disk driver

SYNOPSIS
/dev/dsk/lv*
/dev/rdsk/lv*

DESCRIPTION
The lv devices provide access to disk storage as logical volumes. A logical volume is an entity which behaves like a traditional disk partition, but its storage may span several physical devices.

The constitution of logical volumes known to the system is described in /etc/lvtab, see lvtab(4).

The special files are named according to the convention

/dev/{r}dsk/lv<device-#>

Device numbers may range from 0 to one less than the maximum number of logical volume devices configured in the system. This is 10 by default; this number may be changed by rebuilding a kernel with lboot(1M).

There is a kernel driver, referred to as lv, for the logical volume devices. It is essentially a 'pseudo device' not directly associated with any physical hardware; its function is to map requests on logical volume devices into requests on the underlying disk devices.

(initialization)
Before any logical volume device can be accessed, it must be initialized. This is done by lvinit(1M) which takes information from /etc/lvtab and uses the DIOCSETLV ioctl described below to supply this to the driver.

Note this implies that the root filesystem of a machine may not reside on a logical volume, since lvinit(1M) must be accessible to be run before logical volumes can be used.

ioctl functions
As well as normal read and write operations, the driver supports a number of special io control (ioctl) operations when opened via the character special file in /dev/rdsk. These may be invoked from a user program by the ioctl system call, see ioctl(2). Command values for these are defined in the system include file <sys/dkio.h>.

The ioctl operations are intended for the use of special-purpose utilities. Misuse of them can lead to data corruption, they should be invoked only by the knowledgeable and with extreme caution.
A list of the ioctl commands supported by the lv driver is given below.

**DIOCGETLV**
reads the currently understood constitution of the logical volume device from the driver into a buffer in the calling program. The constitution of a logical volume (i.e., the devices composing it) is described by a 'struct lv' which is defined in `<lv.h>`. The argument to the ioctl call must be a pointer to a 'struct lvioarg', also defined in `<lv.h>`.

The 'lvp' element of this structure must be a pointer to a buffer of sufficient size in the caller's space where the returned information is to be placed.
The 'size' element of the 'lvioarg' structure sets an upper bound to the amount of data which will be returned by the ioctl call. Note that the effective size of a 'struct lv' is variable depending on the number of constituent devices; the amount of data returned will be the minimum of the size of the data held by the driver and the 'size' parameter mentioned above.

If the logical volume device has not yet been initialized, the driver has no information to return. In this case, the ioctl call fails and errno is set to ENODEV.

**DIOCSETLV**
initializes the driver with information about the constitution of the logical volume device. This information controls the mapping done by the logical volume driver from requests on this device to requests on the underlying disk devices.
It consists of a 'struct lv' which is defined in `<lv.h>`. The argument to the ioctl call must be a pointer to a 'struct lvioarg', also defined in `<lv.h>`.

The 'lvp' element of this structure must be a pointer to a buffer in the caller's space containing a 'struct lv' correctly initialized with the desired logical volume description.
The 'size' element of the 'lvioarg' structure determines how much data will be passed to the driver. The effective size of a 'struct lv' is variable depending on the number of constituent devices; the value of the 'size' parameter should be exactly the size of the particular initialized structure.

It is important that the caller be sure that the initialized 'struct lv' does contain a legal and desired logical volume description. The driver performs some consistency checks on the description given
to it; if this is obviously illegal the ioctl call fails and errno is set to EINVAL.

FILES
/dev/dsk/lv*
/dev/rdsk/lv*
/usr/include/sys/dkio.h /usr/include/sys/lv.h

SEE ALSO
lvinit(1M), lvtab(4).
NAME
mem, kmem, mmem – core memory

DESCRIPTION
The file /dev/mem is a special file that is an image of the core memory of the computer. It may be used, for example, to examine, and even to patch the system.

Byte addresses in /dev/mem are interpreted as memory addresses. References to non-existent locations cause errors to be returned.

Examining and patching device registers is likely to lead to unexpected results when read-only or write-only bits are present.

The file /dev/kmem is the same as /dev/mem except that kernel virtual memory rather than physical memory is accessed.

The file /dev/mmem is the same as /dev/kmem when reading and writing. In addition, certain addresses can be mapped into the user’s address space via mmap(2). These addresses will vary depending on the hardware drivers present on any particular system.

FILES
/dev/mem
/dev/kmem
/dev/mmem

WARNING
Some of /dev/kmem cannot be read because of write-only addresses or unequipped memory addresses.
NAME
mouse – optical mouse specifications

DESCRIPTION
Signals: The serial data interface signal level is compatible with RS-423 which has roughly a 10V swing centered about ground. The idle state and true data bits for the interface are Mark level or -5V whereas false data bits and the start bit are spaces or +5V. The serial data is transmitted at 4800 baud with one start bit, eight data bits, and no parity.

Protocol: The mouse provides a five-byte data block whenever there is a change of position or button state. The first byte is a sync byte which has its upper five bits set to 10000 and its lower three bits indicating the button states where a 0 indicates depression. The sync byte looks like this: 10000LMR. The next four bytes contain two difference updates of the mouse’s change in position: X1, Y1, X2, and Y2. Positive values indicate movement to the right or upward. System software ignores bytes beyond the first five until reception of the next sync byte.

Connector Pin Assignment:

<table>
<thead>
<tr>
<th>PIN</th>
<th>SIGNAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>+5V</td>
</tr>
<tr>
<td>9</td>
<td>GND</td>
</tr>
<tr>
<td>5</td>
<td>MTXD</td>
</tr>
<tr>
<td>3</td>
<td>-5V</td>
</tr>
</tbody>
</table>
NAME

mtio – magnetic tape interface

DESCRIPTION

Mtio describes the generic interface provided for dealing with the various types of magnetic tape drives supported on the IRIS-4D. 1/4" streaming cartridge tapes, 1/2" nine-track tapes, and 8 mm video tapes are supported.

Tape drives are accessed through special device files in the /dev/mtl/* and /dev/rmt/* directory. Refer to intro(7) for a general description of the naming conventions for device files in theses subdirectories. Naming conventions for the specific devices are listed under ts(7), tps(7), and xmt(7).

Normally the device specific name is linked to a user friendly synonym for ease of use. Many commands that manipulate magnetic tape refer to these synonyms rather than device specific names. There are up to four user friendly device synonyms:

/dev/tape
This is the tape unit that is the default system tape drive. It is linked to one of the specific device names from ts(7), tps(7) or xmt(7). This device rewinds the tape when closed. For 1/4" cartridge tapes, data bytes are swapped to be compatible with the Iris 2000 and 3000 series machines.

/dev/nrtape
Same as /dev/tape, except the tape is not rewound when closed.

/dev/tapens
Same as /dev/tape, except no byte swapping is done; exists only for 1/4" cartridge tapes.

/dev/nrtapens
Same as /dev/nrtape, except no byte swapping is done; exists only for 1/4" cartridge tapes.

The system makes it possible to treat the tape similar to any other file, with some restrictions on data alignment and request sizes. Seeks do not have their usual meaning and it is not possible to read or write other than a multiple of the fixed block size when using the fixed block device. Writing in very small blocks (less than 4096 bytes) is inadvisable because this tends to consume more tape (create large record gaps verses data) and causes the tape to stop its’ streaming motion.

The standard 1/4" tape consists of a series of 512 byte records terminated by an end-of-file. Other tape devices (such as 9 track and 8 mm) typically support both fixed size block format, and variable size blocks format. (Note: the xmt(7M) driver supports only the variable format.)
When using the variable format, there is an upper limit to the size of a single read or write, typically the size of the RAM buffer on the drive. At this time, the upper limit is 64K bytes on 9 track, and 240K bytes on the 8 mm drives. This information may be obtained by use of the MTIOCGET-BLKNINFO ioctl (for SCSI tape drives only). The main use of this format is to allow small header blocks at the beginning of a tape file, while the rest are typically the same (larger) size.

When the fixed block size device is used, the size of a single read or write request is limited only by physical memory. Currently the default size is 512 bytes on 9 track, and 1024 on 8 mm. This size may be reset with the MTSCSI_SETFIXED ioctl; the value remains set until the next boot or reset via ioctl.

A tape is normally open for reading and/or writing, but a tape cannot be read and written simultaneously. After a rewind, an unload, or an MTAFILE ioctl, writes may follow reads and vice-versa, otherwise only reads, or only writes may be done unless the tape is first closed; performing an MTWEOF ioctl is considered to be a write operation. Whenever the tape is closed after being written to, a file-mark is written (2 on 9 track tapes) unless the tape has been unloaded or rewound just prior to the close.

Some tape drives support more than one speed; for SCSI 9 track tape drives, the default is 100 ips (streaming mode); this may be set to 50 ips by using the MTSCSI_SPEED ioctl. Some tape drives such as the Kennedy 96XX models do auto density select when reading; this can be disabled only by using the drive’s front panel setup mode.

The MTANSI ioctl allows writing of ANSI 3.27 style labels after the early warning point (naturally, drives that don’t support variable sized blocks won’t support 80 byte labels). It is currently implemented only for SCSI tape drives. It remains set until the next close, or reset with a 0 argument. If used, standard SGI EOT handling for tar, bru, and cpio won’t work correctly while set. An arg of 1 enables, 0 disables. NOTE: when the EOT marker is encountered, the current i/o operation returns either a short count (if any i/o completed), or -1 (if no i/o completed); it is the programmers responsibility to determine if EOT has been encountered by using MTIOCGET or logic internal to the program. This ioctl should be issued AFTER encountering EOT, if ANSI 3.27 EOT handling is desired. Subsequent i/o’s will then be allowed, and will return the count actually transferred, or -1 if the drive was unable to transfer any data. The standard calls for writing a FM before the label. If this is not done, the drive may return label info as data.
The ioctl's and the structures they use are defined in the file
/usr/include/sys/mtio.h.

NOTES
Some of the ioctl's are specific to a driver or tape drive; The MTAFILE
ioctl will return EINVVAL if the tape drive is not a 9 track drive, since 1/4"
cartridge and Exabyte drives only allow appending after a filemark.

When the a tape device is opened, the tape is rewound only if there has
been a media change, or the drive has gone offline, or there has been a bus
reset (normally only after a reboot or powerup).

FILES
/dev/tape, /dev/nrtape, /dev/tapens, /dev/nrtapens
/dev/mt/tps*
/dev/rmt/tps*

SEE ALSO
bru(1), cpio(1), mt(1), tar(1), ts(7M), tps(7M), xmt(7M)

ORIGIN
4.3BSD
NAME

networking -- introduction to networking facilities

SYNOPSIS

#include <sys/socket.h>
#include <net/route.h>
#include <net/if.h>

DESCRIPTION

This section briefly describes the networking facilities available in the system. Documentation in this part of section 7 is broken up into three areas: protocol families (domains), protocols, and network interfaces. Entries describing a protocol family are marked "7F," while entries describing protocol use are marked "7P." Hardware support for network interfaces are found among the standard "7" entries.

All network protocols are associated with a specific protocol family. A protocol family provides basic services to the protocol implementation to allow it to function within a specific network environment. These services may include packet fragmentation and reassembly, routing, addressing, and basic transport. A protocol family may support multiple methods of addressing, though the current protocol implementations do not. A protocol family is normally comprised of a number of protocols, one per socket(2) type. It is not required that a protocol family support all socket types. A protocol family may contain multiple protocols supporting the same socket abstraction.

A protocol supports one of the socket abstractions detailed in socket(2). A specific protocol may be accessed either by creating a socket of the appropriate type and protocol family, or by requesting the protocol explicitly when creating a socket. Protocols normally accept only one type of address format, usually determined by the addressing structure inherent in the design of the protocol family/network architecture. Certain semantics of the basic socket abstractions are protocol specific. All protocols are expected to support the basic model for their particular socket type, but may, in addition, provide non-standard facilities or extensions to a mechanism. For example, a protocol supporting the SOCK_STREAM abstraction may allow more than one byte of out-of-band data to be transmitted per out-of-band message.

A network interface is similar to a device interface. Network interfaces comprise the lowest layer of the networking subsystem, interacting with the actual transport hardware. An interface may support one or more protocol families and/or address formats. The ethernet(7) manual entry lists messages which may appear on the console and/or in the system error log, /usr/adm/SYSLOG (see syslogd(1M)), due to errors in device operation.
PROTOCOLS
The system currently supports the DARPA Internet protocols. Raw socket interfaces are provided to the IP protocol layer of the DARPA Internet and to the link-level layer. Consult the appropriate manual pages in this section for more information regarding the support for each protocol family.

ADDRESSING
Associated with each protocol family is an address format. The following address formats are used by the system (and additional formats are defined for possible future implementation):

```c
#define AF_UNIX 1 /* local to host (pipes) */
#define AF_INET 2 /* internetwork: UDP, TCP, etc. */
```

ROUTING
The network facilities provided limited packet routing. A simple set of data structures comprise a "routing table" used in selecting the appropriate network interface when transmitting packets. This table contains a single entry for each route to a specific network or host. A user process, the routing daemon, maintains this data base with the aid of two socket-specific `ioctl(2)` commands, SIOCADDRT and SIOCDELRRT. The commands allow the addition and deletion of a single routing table entry, respectively. Routing table manipulations may only be carried out by super-user.

A routing table entry has the following form, as defined in `<net/route.h>`;

```c
struct rentry {
    u_long rt_hash;
    struct sockaddr rt_dst;
    struct sockaddr rt_gateway;
    short rt_flags;
    short rt_refcnt;
    u_long rt_use;
    struct ifnet *rt_ifp;
};
```

with `rt_flags` defined from,

```c
#define RTF_UP 0x1 /* route usable */
#define RTF_GATEWAY 0x2 /* destination is a gateway */
#define RTF_HOST 0x4 /* host entry (net otherwise) */
#define RTF_DYNAMIC 0x10 /* created dynamically (by redirect) */
#define RTF_MODIFIED 0x10 /* modified dynamically (by redirect) */
```
Routing table entries come in three flavors: for a specific host, for all hosts on a specific network, for any destination not matched by entries of the first two types (a wildcard route). When the system is booted and addresses are assigned to the network interfaces, each protocol family installs a routing table entry for each interface when it is ready for traffic. Normally the protocol specifies the route through each interface as a "direct" connection to the destination host or network. If the route is direct, the transport layer of a protocol family usually requests the packet be sent to the same host specified in the packet. Otherwise, the interface is requested to address the packet to the gateway listed in the routing entry (i.e., the packet is forwarded).

Routing table entries installed by a user process may not specify the hash, reference count, use, or interface fields; these are filled in by the routing routines. If a route is in use when it is deleted (rt_refcnt is non-zero), the routing entry will be marked down and removed from the routing table, but the resources associated with it will not be reclaimed until all references to it are released. The routing code returns EEXIST if requested to duplicate an existing entry, ESRCH if requested to delete a non-existent entry, or ENOBUFFS if insufficient resources were available to install a new route. User processes read the routing tables through the /dev/kmem device. The rt_use field contains the number of packets sent along the route.

When routing a packet, the kernel will first attempt to find a route to the destination host. Failing that, a search is made for a route to the network of the destination. Finally, any route to a default ("wildcard") gateway is chosen. If multiple routes are present in the table, the first route found will be used. If no entry is found, the destination is declared to be unreachable.

A wildcard routing entry is specified with a zero destination address value. Wildcard routes are used only when the system fails to find a route to the destination host and network. The combination of wildcard routes and routing redirects can provide an economical mechanism for routing traffic.

INTERFACES

Each network interface in a system corresponds to a path through which messages may be sent and received. A network interface usually has a hardware device associated with it, though certain interfaces such as the loopback interface, lo(7), do not.

The following ioctl calls may be used to manipulate network interfaces. The ioctl is made on a socket (typically of type SOCK_DGRAM) in the desired domain. Unless specified otherwise, the request takes an ifrequest structure as its parameter. This structure has the form
struct ifreq {
    char ifr_name[IFNAMSIZ]; /* if name, e.g. "enp0" */
    union {
        struct sockaddr ifru_addr;
        struct sockaddr ifru_dstaddr;
        struct sockaddr ifru_broadaddr;
        short ifru_flags;
        int ifru_metric;
        caddr_t ifru_data;
    } ifr_ifru;
}
#define ifr_addr ifr_ifru.ifru_addr /* address */
#define ifr_dstaddr ifr_ifru.ifru_dstaddr /* other end of p-to-p link */
#define ifr_broadaddr ifr_ifru.ifru_broadaddr /* broadcast address */
#define ifr_flags ifr_ifru.ifru_flags /* flags */
#define ifr_metric ifr_ifru.ifru_metric /* metric */
#define ifr_data ifr_ifru.ifru_data /* for use by interface */

SIOCSIFADDR
Set interface address for protocol family. Following the address assignment, the "initialization" routine for the interface is called.

SIOCGIFADDR
Get interface address for protocol family.

SIOCSIFDSTADDR
Set point to point address for protocol family and interface.

SIOCGIFDSTADDR
Get point to point address for protocol family and interface.

SIOCSIFBRDADDR
Set broadcast address for protocol family and interface.

SIOCGIFBRDADDR
Get broadcast address for protocol family and interface.

SIOCSIFFLAGS
Set interface flags field. If the interface is marked down, any processes currently routing packets through the interface are notified; some interfaces may be reset so that incoming packets are no longer received. When marked up again, the interface is reinitialized.
**SIOSIFCONF**

Get interface configuration list. This request takes an `ifconf` structure (see below) as a value-result parameter. The `ifc_len` field should be initially set to the size of the buffer pointed to by `ifc_buf`. On return it will contain the length, in bytes, of the configuration list.

```c
/*
 * Structure used in SIOSIFCONF request.
 * Used to retrieve interface configuration
 * for machine (useful for programs which
 * must know all networks accessible).
 */
struct ifconf {
    int ifc_len; /* size of associated buffer */
    union {
        caddr_t ifc_buf;
        struct ifreq *ifc_req;
    } ifc_ifcu;
#define ifc_buf ifc_ifcu.ifc_ifcu.ifc_buf /* buffer address */
#define ifc_req ifc_ifcu.ifc_ifcu.ifc_req /* array of structures returned */
};
```

**SEE ALSO**

socket(2), ioctl(2), routed(1M)
NAME
  null – the null file

DESCRIPTION
  Data written on the null special file, /dev/null, is discarded.
  Reads from a null special file always return 0 bytes.

FILES
  /dev/null
NAME
plp – parallel line printer interface

SYNOPSIS
#include <sys/plp.h>
/dev/plp

DESCRIPTION
The special file /dev/plp refers to the parallel printer interface on the IRIS
4D/20. The plp device supports output to a Centronics-compatible printer
connected to the builtin parallel printer port. Normally, /dev/plp is directly
accessed only by a print spooling mechanism such as the lp(1) subsystem.

The special file /dev/plp may only be open for writing by one process at a
time. However, several processes may open the device read-only so as to
obtain printer status. A printer reset (INPUT PRIME) is issued whenever
the device is opened for writing.

The following iocltls are defined in the include file <sys/plp.h> and may be
used to control the device:

PLPIOCSTATUS
Returns the current printer status.

PLPIOCRESET
Cancels any current printing and
asserts the "INPUT PRIME" signal to
the printer.

The PLPIOCSTATUS ioctl returns the printer status as an integer bit-mask
based on constants defined in <sys/plp.h>. The constants are described
below:

BUSY The printer is currently printing
FAULT Printer is in fault state
EOI End of Ink (Ribbon out)
EOP End of Paper
ONLINE Printer is on line

Not all printers support all of these status indications.

FILES
/dev/plp

SEE ALSO
lp(1)
NAME
prf – operating system profiler

DESCRIPTION
The special file /dev/prf provides access to activity information in the operating system. Writing the file loads the measurement facility with text addresses to be monitored. Reading the file returns these addresses and a set of counters indicative of activity between adjacent text addresses.

The recording mechanism is driven by a separate profiling clock. The profiling clock period is set so that it does not become synchronized with the regular system clock. Samples that catch the operating system are matched against the stored text addresses and increment corresponding counters for later processing.

The file /dev/prf is a pseudo-device with no associated hardware.

FILES
/dev/prf

SEE ALSO
profiler(1M).
NAME

pty – pseudo terminal driver

DESCRIPTION

The pty driver provides a device-pair termed a pseudo terminal. A pseudo terminal is a pair of character devices, a master device and a slave device. The slave device provides processes an interface identical to that described in termio(7). However, whereas all other devices which provide the interface described in termio(7) have a hardware device of some sort behind them, the slave device has, instead, another process manipulating it through the master half of the pseudo terminal. That is, anything written on the master device is given to the slave device as input and anything written on the slave device is presented as input on the master device.

The following ioctl calls apply only to pseudo terminals:

TIOCPKT

Enable/disable packet mode. Packet mode is enabled by specifying (by reference) a nonzero parameter and disabled by specifying (by reference) a zero parameter. When applied to the master side of a pseudo terminal, each subsequent read from the terminal will return data written on the slave part of the pseudo terminal preceded by a zero byte (symbolically defined as TIOCPKT_DATA), or a single byte reflecting control status information. In the latter case, the byte is an inclusive-or of zero or more of the bits:

TIOCPKT_FLUSHREAD
    whenever the read queue for the terminal is flushed.

TIOCPKT_FLUSHWRITE
    whenever the write queue for the terminal is flushed.

TIOCPKT_STOP
    whenever output to the terminal is stopped a la "S.

TIOCPKT_START
    whenever output to the terminal is restarted.

TIOCPKT_DOSTOP
    whenever t_stopc is "S and t_startc is "Q.

TIOCPKT_NOSTOP
    whenever the start and stop characters are not "S/"Q.

This mode is used by rlogin(1C) and rlogin(1M) to implement a remote-echoed, locally "S/"Q flow-controlled remote login with proper back-flushing of output; it can be used by other similar programs.
ALLOCATION

The code sequence shown below demonstrates how to allocate pseudo terminals. Note that pseudo terminals, like all files, must have the correct file permissions to be accessible.

```c
#include <fcntl.h>
#include <sys/types.h>
#include <sys/sysmacros.h>
#include <sys/stat.h>
#include <stdio.h>

/*
 * Find a pseudo tty to use. Fill in "name" with the name of the tty.
 * Return the open descriptor for the controlling side. Caller is
 * responsible for opening the slave side, if it wants to.
 */
int
findPseudoTTY(name, ptynum)
    char *name;
    int *ptynum;
{
    int fd;
    struct stat sb;

    fd = open("/dev/ptc", O_RDWR|O_NDELAY);
    if (fd >= 0) {
        if (fstat(fd, &sb) < 0) {
            close(fd);
            return -1;
        }
        *ptynum = minor(sb.st_rdev);
        printf(name, "/dev/ttyq%d", *ptynum);
    }
    return fd;
}
FILES
/dev/ptc - master pseudo terminal
/dev/ttyq[0-99] - slave pseudo terminals
NAME
raw – raw network protocol family

SYNOPSIS
#include <sys/types.h>
#include <net/raw.h>

DESCRIPTION
The Raw protocol family is a collection of input decapsulation protocols layered atop the data link protocol of a network interface. The Raw family supports only the SOCK_RAW socket type.

Addressing
Sockets bound to the Raw protocol family use the following addressing structure, defined in <net/raw.h>:

```c
struct sockaddr_raw {
    u_short sr_family;
    union {
        struct {
            u_short sr_port;
            char sr_ifname[RAW_IFNAMSIZ];
        };
        struct {
            char sr_addr[RAW_MAXADDRLEN];
        };
    }
};
```

The address format differs between local and foreign usage. A local AF_RAW sockaddr contains a port, identifying the socket to which this address is bound, and the zero-padded name of a network interface. If the address to bind contains a zeroed interface name, the primary interface is used. Port numbering depends on the protocol of the socket being bound. A foreign AF_RAW sockaddr contains a link-layer destination address.

Protocols
There are two protocols in the Raw family, Snoop and Drain. The Snoop protocol captures link-layer packets which match a bitfield filter and transports them to that filter’s socket. Snoop prepends a header containing packet state, sequence, and a timestamp. The Drain protocol receives packets having network-layer type codes or encapsulations not implemented by the kernel.

Both protocols transmit packets with a link-layer header fetched from the beginning of the user’s write(2) or send(2) buffer. They ignore any destination address supplied to sendto(2) or connect(2). However, connecting restricts input to packets originating from the connected address.
On input, a protocol-specific header, the link-layer header, and packet data are copied to the user’s `read(2)` or `recv(2)` buffer. All raw domain protocols guarantee that the offset of packet data from the beginning of the user’s buffer is congruent with `RAW_ALIGNNGRAIN`. `RAW_HDRPAD(hdrsize)` yields the byte-padding needed to align packet data, given the link-layer header’s size in bytes.

To gather Raw protocol family statistics, call `ioctl(2)` with a Snoop or Drain socket, the `SIOCRAWSTATS` command, and the address of the following structure, defined in `<net/rraw.h>`:

```c
struct rawstats {
    struct snoopstats {
        u_long ss_seq;
        u_long ss_ifdops;
        u_long ss_sbdrops;
    } rs_snoop;
    struct drainstats {
        u_long ds_ifdops;
        u_long ds_sbdrops;
    } rs_drain;
};
```

The `ss_seq` structure member tells the current Snoop sequence number, which counts all packets received by the hardware, whether or not they can be decapsulated. The `ifdops` members tell how many packets were dropped by the network interface due to resource shortages or hardware errors. The `sbdrops` members tell how many packets were decapsulated by the network interface but dropped due to socket buffer limits. See `getsockopt(2)` for information on socket buffer sizes and how to change them.

SEE ALSO

socket(2), snoop(7P), drain(7P)
NAME

root, root, usr, rusr, swap, rswap – Partition names.

DESCRIPTION

These are the device special files providing access to important partitions on the root disk drive of the system. The names beginning with ‘r’ are the raw or ‘character’ device access, the others are the block device access which uses the kernel buffer system.

The standard partition allocation by Silicon Graphics has root on partition 0, swap on partition 1, and /usr on partition 6. Partition 7 maps the entire usable portion of the disk (excluding the defect area and volume header). Partition 8 maps the volume header (see vh(7M), prvtoc(1M), dvhtool(1M)). Partition 9 maps the defect area, (where applicable; there is no defect area on SCSI drives). Partition 10 maps the entire drive.

The standard IRIS-4D with ESDI drives has /dev/root linked to /dev/dsk/ips0d0s0, /dev/swap linked to /dev/dsk/ips0d0s1, and /dev/usr linked to /dev/dsk/ips0d0s6.

The standard IRIS-4D with SCSI drives has /dev/root linked to /dev/dsk/dks0d1s0, /dev/swap linked to /dev/dsk/dks0d1s1, and /dev/usr linked to /dev/dsk/dks0d1s6.

FILES

/dev/dsk/ips*
/dev/rdsk/ips*
/dev/dsk/dks*
/dev/rdsk/dks*
/dev/root
/dev/usr
/dev/swap

SEE ALSO

dvhtool(1M), prvtoc(1M), fx(1M), vh(7M).
NAME

SA – devices administered by System Administration

DESCRIPTION

The files in the directories /dev/SA (for block devices) and the /dev/rSA (for raw devices) are used by System Administration to access the devices on which it operates. For devices that support more than one partition (like disks) the /dev/(r)SA entry is linked to the partition that spans the entire device. Not all /dev/(r)SA entries are used by all System Administration commands.

FILES

/dev/SA
/dev/rSA

SEE ALSO

NAME
smfd – SCSI floppy disk driver

SYNOPSIS
/dev/rdsk/fds*

DESCRIPTION
This driver supports 5 ¼" floppy drives in 3 standard formats when used with the freestanding SCSI floppy drive, and for the Konica K-510 10.7 Mbyte internal floppy drive. The standard single and double density dual sided 3 ½" drive is also supported 44Mbyte capacities.

The driver also supports a large range of special formats on the non-Konica drives by use of the SMFDSETMODE ioctl (defined along with the associated structure in /usr/include/sys/smfd.h).

Most of the ioctls for the dksed also apply to this driver, with the exception of the bad block handling. The volume header ioctl DIOCGETVH is supported by creating the structure based on the current driver parameters. The DIOCGETVH ioctl is not supported. Individual tracks may be formatted on the non-Konica devices using the DIOCFMTTRK ioctl and the fmt_map_info structure defined in /usr/include/sys/dvh.h.

The standard devices (and suffixes) are the standard PC formats of 360Kb (.48), 720Kb (.96), 1.2 Mb (.96hi) on the regular 5 ¼" device, the 10.7 Mb on the Konica K-510 drive (.480), and the standard 720 Kb (.3.5), and 1.44Mb (.3.5hi) for the 3 ½" device.

The special files are named according to the convention discussed in intro(7M), with the exception of .type, instead of s#:

/dev/[r]dsk/fds<controller->d<drive->.type

These devices are not supported for filesystem use, however they may be used as backup devices, and for data interchange with other systems.

Ordinarily, the devices may be opened by more than one process at a time, as long as all are opened at the same density. Attempts to open at a different density than which it is already open will fail. If the first open(2) of a floppy device uses the O_EXCL option, then no other process may open the drive. In addition, this option will allow the open to complete successfully even if there is no floppy in the drive; otherwise the open fails.

FILES
/dev/[r]dsk/fds?d?.48
/dev/[r]dsk/fds?d?.96
/dev/[r]dsk/fds?d?.96hi
/dev/[r]dsk/fds?d?.3.5
SMFD(7M) Silicon Graphics SMFD(7M)

/dev/[r]dsk/fds?d?:3.5hi
/dev/[r]dsk/fds?d?:480

SEE ALSO
fx(1M), softpc(1), intro(7), dksc(7M), open(2)
NAME
snoop – network monitoring protocol

SYNOPSIS
#include <sys/types.h>
#include <net/raw.h>

s = socket(PF_RAW, SOCK_RAW, RAWPROTO_SNOOP);

DESCRIPTION
The Snoop protocol provides promiscuous packet capture with filtering. It treats packets as datagrams containing a link-layer header followed by data. Snoop uses the Raw address format, assigning a unique port to a socket bound to port zero, otherwise binding the specified port if it is valid. Valid ports range from SNOOP_MINPORT to SNOOP_MAXPORT.

Snoop associates a set of SNOOP_MAXFILTERS packet filters with each network interface. Each filter contains an array of mask bits, a parallel array of bits to match against the masked packet’s bits, the filter’s index in the interface’s filter table, and a port identifying the socket which added this filter.

struct snoopfilter {
    u_long sf_mask[SNOOP_FILTERLEN];
    u_long sf_match[SNOOP_FILTERLEN];
    u_short sf_allocated:1,
        sf_active:1,
        sf_promisc:1,
        sf_allmulti:1,
        sf_index:SNOOP_MAXFILTSHIFT;
    u_short sf_port;
};

The mask is applied to at most SNOOP_FILTERLEN long integers of the packet. If the link-layer header size is not congruent with RAW_ALIGNGRAIN, the mask and match arrays begin with RAW_HDRPAD(hdrsize) bytes of padding, in order to preserve native addressability of packet data. The RAW_HDR(addr, hdrtype) macro adjusts and coerces a RAW_ALIGNGRAIN-congruent address into a pointer to the packet header type. Use RAW_HDR to convert sf_mask and sf_match into appropriately padded, typed pointers. Only sf_mask and sf_match may be effectively set by user code.

Call ioctl(2) on a bound Snoop socket with the SIOCADDNSNOOP command and the address of a snoopfilter, to add a filter. The SIOCDELSNOOP ioctl command takes the address of an integer telling the index of a filter to delete. The SIOCSNOOPLEN command takes the address of an integer
telling how many bytes of packet data to capture (the link-layer header is always captured). By default, all received bytes of packet data are captured. The SIOCXSNOOPING command takes the address of an integer boolean telling whether to start or stop capture.

The SIOCERRSNOOP ioctl command establishes an error filter. It takes the address of an integer containing error flag bits (see below), and designates the socket being operated on as the error snooper. There may be at most one error snooper per network interface. Only packets received with errors indicated by bits in the integer argument will be captured.

Snoop applies filters to a non-erroneous packet in index order, matching all filters against the packet. It then prepends the following header to the alignment-padded, link-layer header:

```c
struct snoopheader {
    u_long    snoop_seq;
    u_short   snoop_flags;
    u_short   snoop_packetlen;
    struct timeval snoop_timestamp;
};
```

A snoopheader contains a reception sequence number, packet state flags, length in bytes of the link-layer packet excluding frame check and preamble, and a reception timestamp. The bits in snoop_flags describe the packet’s state as follows:

- **SN_PROMISC** packet was not destined for this interface
- **SN_ERROR** receive error specified by the following bits:
  - **SNERR_FRAME** packet received, but with framing error
  - **SNERR_CHECKSUM** packet received, but with CRC error
  - **SNERR_TOOBIG** packet received, truncated to fit buffer
  - **SNERR_TOOSMALL** packet may be received, size less than minimum
  - **SNERR_NOBUFS** no packet received, out of buffers
  - **SNERR_OVERFLOW** no packet received, input silo overflow
  - **SNERR_MEMORY** no packet received, buffer memory error

The snoop_timestamp member contains the packet’s reception time, with precision limited by the operating system’s clock tick parameter (see times(2)).

Output on a Snoop socket, using write(2) or send(2), takes a buffer address pointing at the link-layer packet to be transmitted. Output buffers may need to begin with RAW_HDRPADD bytes of padding to ensure addressability of structured data, but such padding is not passed to write.
EXAMPLES
To capture all packets from an Ethernet network interface, first declare an
input buffer structure:

```c
#include <sys/types.h>
#include <net/raw.h>
#include <netinet/if_ether.h>

#define ETHERHDRPAD RAW_HDRPAD(sizeof(struct ether_header))

struct etherpacket {
    struct snoopheader snoop;
    char pad[ETHERHDRPAD];
    struct ether_header ether;
    char data[ETHERMTU];
};
```

Then create a Snoop socket (error handling is omitted for clarity). Bind it
to the desired interface, e.g., `en0` (to bind to the primary interface, zero
`sr_ifname`):

```c
int s;
struct sockaddr_raw sr;

s = socket(PF_RAW, SOCK_RAW, RAWPROTO_SNOOP);
sr.sa_family = AF_RAW;
sr.sa_port = 0;
strcpy(sr.sa_ifname, "en0", sizeof sr.sa_ifname);
bind(s, &sr, sizeof sr);
```

Initialize a filter with no mask bits set, in order to match all packets. Add it
to the interface’s filter set:

```c
struct snoopfilter sf;

bzero((char *) &sf, sizeof sf);
ioctl(s, SIOCADDSSNOOP, &sf);
```

Increase the socket’s receive buffer size to a generous upper bound, to cope
with promiscuous reception of heavy traffic. Turn snooping on and read
captured packets:
struct etherpacket ep;
int cc = 60000, on = 1;

setsockopt(s, SOL_SOCKET, SO_RCVBUF, (char *)&cc, sizeof cc);
ioctl(s, SIOCSNOOPING, &on);
for (;;) {
    cc = read(s, (char *)&ep, sizeof ep);
    /* .. . */
}

To capture ARP packets from a specific Ethernet host, first declare an appropriate input buffer structure:

#include <sys/types.h>
#include <net/raw.h>
#include <net/if_arp.h>
#include <netinet/if_ether.h>

#define ETHERHDRPAD RAW_HDRPAD(sizeof(struct ether_header))

struct arp_packet {
    struct snoopheader snoop;
    char pad[ETHERHDRPAD];
    struct ether_header ether;
    struct ether_arp arp;
};

Create and bind a Snoop socket as shown in the previous example. Then initialize a filter to capture all ARP requests originating from a remote_ether_addr:

struct snoopfilter sf;
struct ether_header *eh;

bzero((char *)&sf, sizeof sf);
eh = RAW_HDR(sf.sf_mask, struct ether_header);
memset(eh->ether_dhost, 0xff, sizeof eh->ether_dhost);
 eh->ether_type = 0xffff;
 eh = RAW_HDR(sf.sf_match, struct ether_header);
bcopy(remote_ether_addr, eh->ether_dhost, sizeof eh->ether_dhost);
 eh->ether_type = htons(ETHERTYPE_ARP);

Finally, add the filter and start capturing packets:
struct arp_packet ap;
int cc, on = 1;

ioctl(s, SIOCADDSENSPOOL, &sf);
ioctl(s, SIOCSNIPPING, &on);
for (;;) {
    cc = read(s, (char *) &ap, sizeof ap);
    /* . . . */
}

DIAGNOSTICS
A socket operation may fail with one of the following errors returned:

[EISCONN] when trying to establish a connection on a socket which already has one, or when trying to send a datagram with the destination address specified and the socket is already connected;

[ENOBUFS] when the system runs out of memory for an internal data structure or a send or receive buffer;

[EADDRINUSE] when an attempt is made to create a socket with a port which has already been allocated;

[EADDRNOTAVAIL] when an attempt is made to bind an address naming a non-existent network interface to a Raw family socket.

[EOPNOTSUPP] when an ioctl operation not supported by the Snoop protocol is attempted.

[EINVAL] when a Snoop ioctl argument is out of bounds or otherwise invalid.

[EBUSY] when an error snooper is running and another process attempts to set an error filter with SIOCERRSNOOP.

SEE ALSO
getsockopt(2), socket(2), intro(3), ethernet(7), raw(7P), drain(7P)
NAME
streamio – STREAMS ioctl commands

SYNOPSIS

#include <stropts.h>
int ioctl (fdles, command, arg)
int fdles, command;

DESCRIPTION

STREAMS [see intro(2)] ioctl commands are a subset of ioctl(2) system calls which perform a variety of control functions on streams. The arguments command and arg are passed to the file designated by fdles and are interpreted by the stream head. Certain combinations of these arguments may be passed to a module or driver in the stream.

fdles is an open file descriptor that refers to a stream. command determines the control function to be performed as described below. arg represents additional information that is needed by this command. The type of arg depends upon the command, but it is generally an integer or a pointer to a command-specific data structure.

Since these STREAMS commands are a subset of ioctl, they are subject to the errors described there. In addition to those errors, the call will fail with errno set to EINVAL, without processing a control function, if the stream referenced by fdles is linked below a multiplexor, or if command is not a valid value for a stream.

Also, as described in ioctl, STREAMS modules and drivers can detect errors. In this case, the module or driver sends an error message to the stream head containing an error value. This causes subsequent system calls to fail with errno set to this value.

COMMAND FUNCTIONS

The following ioctl commands, with error values indicated, are applicable to all STREAMS files:

I_PUSH

Pushes the module whose name is pointed to by arg onto the top of the current stream, just below the stream head. It then calls the open routine of the newly_PUSHed module. On failure, errno is set to one of the following values:

[EINVAL] Invalid module name.

[EFAULT] arg points outside the allocated address space.

[ENXIO] Hangup received on fildes.

I_POP
Removes the module just below the stream head of the stream pointed to by fildes. arg should be 0 in an I_POP request. On failure, errno is set to one of the following values:

[EINVAL] No module present in the stream.

[ENXIO] Hangup received on fildes.

I_LOOK
Retrieves the name of the module just below the stream head of the stream pointed to by fildes, and places it in a null terminated character string pointed at by arg. The buffer pointed to by arg should be at least FMAMESZ+1 bytes long. An #include <sys/conf.h> declaration is required. On failure, errno is set to one of the following values:

[EFAULT] arg points outside the allocated address space.

[EINVAL] No module present in stream.

I_FLUSH
This request flushes all input and/or output queues, depending on the value of arg. Legal arg values are:

FLUSHR Flush read queues.

FLUSHW Flush write queues.

FLUSHRW Flush read and write queues.

On failure, errno is set to one of the following values:

[ENOSR] Unable to allocate buffers for flush message due to insufficient STREAMS memory resources.

[EINVAL] Invalid arg value.

[ENXIO] Hangup received on fildes.

I_SETSIG
Informs the stream head that the user wishes the kernel to issue the SIGPOLL signal [see signal(2) and sigset(2)] when a particular event has occurred on the stream associated with fildes. I_SETSIG supports an asynchronous processing capability in STREAMS. The value of arg is a bitmask that specifies the events for which the user should be signaled. It is the bitwise-OR of any combination of the following constants:
S_INPUT  A non-priority message has arrived on a stream head read queue, and no other messages existed on that queue before this message was placed there. This is set even if the message is of zero length.

S_HIPRI  A priority message is present on the stream head read queue. This is set even if the message is of zero length.

S_OUTPUT  The write queue just below the stream head is no longer full. This notifies the user that there is room on the queue for sending (or writing) data downstream.

S_MSG  A STREAMS signal message that contains the SIGPOLL signal has reached the front of the stream head read queue.

A user process may choose to be signaled only of priority messages by setting the arg bitmask to the value S_HIPRI.

Processes that wish to receive SIGPOLL signals must explicitly register to receive them using I_SETSIG. If several processes register to receive this signal for the same event on the same Stream, each process will be signaled when the event occurs.

If the value of arg is zero, the calling process will be unregistered and will not receive further SIGPOLL signals. On failure, errno is set to one of the following values:

[EINVAL]  arg value is invalid or arg is zero and process is not registered to receive the SIGPOLL signal.

[EAGAIN]  Allocation of a data structure to store the signal request failed.

I_GETSIG  Returns the events for which the calling process is currently registered to be sent a SIGPOLL signal. The events are returned as a bitmask pointed to by arg, where the events are those specified in the description of I_SETSIG above. On failure, errno is set to one of the following values:

[EINVAL]  Process not registered to receive the SIGPOLL signal.
[EFAULT]  arg points outside the allocated address space.

I_FIND

Compared the names of all modules currently present in the stream to the name pointed to by arg, and returns 1 if the named module is present in the stream. It returns 0 if the named module is not present. On failure, errno is set to one of the following values:

[EFAULT]  arg points outside the allocated address space.

[EINVAL]  arg does not contain a valid module name.

I_PEEK

Allows a user to retrieve the information in the first message on the stream head read queue without taking the message off the queue. arg points to a strpeek structure which contains the following members:

```
struct strbuf  ctlbuf;
struct strbuf  databuf;
long           flags;
```

The maxlen field in the ctlbuf and databuf strbuf structures [see getmsg(2)] must be set to the number of bytes of control information and/or data information, respectively, to retrieve. If the user sets flags to RS_HIPRI, I_PEEK will only look for a priority message on the stream head read queue.

I_PEEK returns 1 if a message was retrieved, and returns 0 if no message was found on the stream head read queue, or if the RS_HIPRI flag was set in flags and a priority message was not present on the stream head read queue. It does not wait for a message to arrive. On return, ctlbuf specifies information in the control buffer, databuf specifies information in the data buffer, and flags contains the value 0 or RS_HIPRI. On failure, errno is set to the following value:

[EFAULT]  arg points, or the buffer area specified in ctlbuf or databuf is, outside the allocated address space.

[EBADMSG]  Queued message to be read is not valid for I_PEEK
I_SRDOPT

Sets the read mode using the value of the argument arg. Legal arg values are:

- RNORM   Byte-stream mode, the default.
- RMSGD   Message-discard mode.
- RMSGN   Message-nondiscard mode.

Read modes are described in read(2). On failure, errno is set to the following value:

[EINVAL]   arg is not one of the above legal values.

I_GRDOPT

Returns the current read mode setting in an int pointed to by the argument arg. Read modes are described in read(2). On failure, errno is set to the following value:

[EFAULT]   arg points outside the allocated address space.

I_NREAD

Counts the number of data bytes in data blocks in the first message on the stream head read queue, and places this value in the location pointed to by arg. The return value for the command is the number of messages on the stream head read queue. For example, if zero is returned in arg, but the ioctl return value is greater than zero, this indicates that a zero-length message is next on the queue. On failure, errno is set to the following value:

[EFAULT]   arg points outside the allocated address space.

I_FDINSERT

Creates a message from user specified buffer(s), adds information about another stream and sends the message downstream. The message contains a control part and an optional data part. The data and control parts to be sent are distinguished by placement in separate buffers, as described below.

arg points to a strfdinsert structure which contains the following members:

- struct strbuf   ctilbuf;
- struct strbuf   databuf;
- long            flags;
- int             fildes;
- int             offset;

The len field in the ctilbuf strbuf structure [see putmsg(2)] must be set to the size of a pointer plus the number of bytes
of control information to be sent with the message. *fildes* in the *strfinsert* structure specifies the file descriptor of the other stream. *offset*, which must be word-aligned, specifies the number of bytes beyond the beginning of the control buffer where I_FDISPERT will store a pointer. This pointer will be the address of the read queue structure of the driver for the stream corresponding to *fildes* in the *strfinsert* structure. The *len* field in the *databuf* *strbuf* structure must be set to the number of bytes of data information to be sent with the message or zero if no data part is to be sent.

*flags* specifies the type of message to be created. A non-priority message is created if *flags* is set to 0, and a priority message is created if *flags* is set to RS_HIPRI. For non-priority messages, I_FDISPERT will block if the stream write queue is full due to internal flow control conditions. For priority messages, I_FDISPERT does not block on this condition. For non-priority messages, I_FDISPERT does not block when the write queue is full and O_NDELAY is set. Instead, it fails and sets *errno* to EAGAIN.

I_FDISPERT also blocks, unless prevented by lack of internal resources, waiting for the availability of message blocks in the stream, regardless of priority or whether O_NDELAY has been specified. No partial message is sent. On failure, *errno* is set to one of the following values:

- **[EAGAIN]** A non-priority message was specified, the O_NDELAY flag is set, and the stream write queue is full due to internal flow control conditions.
- **[ENOSR]** Buffers could not be allocated for the message that was to be created due to insufficient STREAMS memory resources.
- **[EFAULT]** *arg* points, or the buffer area specified in *clbuf* or *databuf* is, outside the allocated address space.
- **[EINVAL]** One of the following: *fildes* in the *strfinsert* structure is not a valid, open stream file descriptor; the size of a pointer plus *offset* is greater than the *len* field for the buffer specified through *clptr*; *offset* does not specify a properly-aligned location in the data buffer; an undefined value is stored in
flags.

[ENXIO] Hangup received on fildes of the ioctl call or fildes in the strfdinsert structure.

[ERANGE] The len field for the buffer specified through databuf does not fall within the range specified by the maximum and minimum packet sizes of the topmost stream module, or the len field for the buffer specified through databuf is larger than the maximum configured size of the data part of a message, or the len field for the buffer specified through cllbuf is larger than the maximum configured size of the control part of a message.

I_FDINSERT can also fail if an error message was received by the stream head of the stream corresponding to fildes in the strfdinsert structure. In this case, errno will be set to the value in the message.

I_STR

Constructs an internal STREAMS ioctl message from the data pointed to by arg, and sends that message downstream.

This mechanism is provided to send user ioctl requests to downstream modules and drivers. It allows information to be sent with the ioctl, and will return to the user any information sent upstream by the downstream recipient. I_STR blocks until the system responds with either a positive or negative acknowledgement message, or until the request "times out" after some period of time. If the request times out, it fails with errno set to ETIME.

At most, one I_STR can be active on a stream. Further I_STR calls will block until the active I_STR completes at the stream head. The default timeout interval for these requests is 15 seconds. The O_NDELAY [see open(2)] flag has no effect on this call.

To send requests downstream, arg must point to a strioctl structure which contains the following members:

```c
int ic_cmd;      /* downstream command */
int ic_timeout;  /* ACK/NAK timeout */
int ic_len;      /* length of data arg */
char *ic_dp;     /* ptr to data arg */
```
ic_cmd is the internal ioctl command intended for a down-stream module or driver and ic_timeout is the number of seconds (-1 = infinite, 0 = use default, >0 = as specified) an I_STR request will wait for acknowledgement before timing out. ic_len is the number of bytes in the data argument and ic_dp is a pointer to the data argument. The ic_len field has two uses: on input, it contains the length of the data argument passed in, and on return from the command, it contains the number of bytes being returned to the user (the buffer pointed to by ic_dp should be large enough to contain the maximum amount of data that any module or the driver in the stream can return).

The stream head will convert the information pointed to by the strioctl structure to an internal ioctl command message and send it downstream. On failure, errno is set to one of the following values:

- [ENOSR] Unable to allocate buffers for the ioctl message due to insufficient STREAMS memory resources.
- [EFAULT] arg points, or the buffer area specified by ic_dp and ic_len (separately for data sent and data returned) is, outside the allocated address space.
- [EINVAL] ic_len is less than 0 or ic_len is larger than the maximum configured size of the data part of a message or ic_timeout is less than -1.
- [ENXIO] Hangup received on fildes.
- [ETIME] A downstream ioctl timed out before acknowledgement was received.

An I_STR can also fail while waiting for an acknowledgement if a message indicating an error or a hangup is received at the stream head. In addition, an error code can be returned in the positive or negative acknowledgement message, in the event the ioctl command sent downstream fails. For these cases, I_STR will fail with errno set to the value in the message.
I_SENDFD

Requests the *stream* associated with *fildes* to send a message, containing a file pointer, to the *stream head* at the other end of a *stream* pipe. The file pointer corresponds to *arg*, which must be an integer file descriptor.

I_SENDFD converts *arg* into the corresponding system file pointer. It allocates a message block and inserts the file pointer in the block. The user id and group id associated with the sending process are also inserted. This message is placed directly on the read queue [see intro(2)] of the *stream head* at the other end of the *stream* pipe to which it is connected. On failure, *errno* is set to one of the following values:

[EAGAIN] The sending *stream* is unable to allocate a message block to contain the file pointer.

[EAGAIN] The read queue of the receiving *stream head* is full and cannot accept the message sent by I_SENDFD.

[EBADF] *arg* is not a valid, open file descriptor.

[EINVAL] *fildes* is not connected to a *stream* pipe.

[ENXIO] Hangup received on *fildes*.

I_RECVFD

Retrieves the file descriptor associated with the message sent by an I_SENDFD *ioctl* over a *stream* pipe. *arg* is a pointer to a data buffer large enough to hold an *strrecvfd* data structure containing the following members:

```
int fd;
unsigned short uid;
unsigned short gid;
char fill[8];
```

*fd* is an integer file descriptor. *uid* and *gid* are the user id and group id, respectively, of the sending *stream*.

If O_NDELAY is not set [see open(2)], I_RECVFD will block until a message is present at the *stream head*. If O_NDELAY is set, I_RECVFD will fail with *errno* set to EAGAIN if no message is present at the *stream head*.

If the message at the *stream head* is a message sent by an I_SENDFD, a new user file descriptor is allocated for the file pointer contained in the message. The new file descriptor is placed in the *fd* field of the *strrecvfd* structure. The structure is copied into the user data buffer pointed to by *arg*. 
On failure, *errno* is set to one of the following values:

- **[EAGAIN]** A message was not present at the *stream head* read queue, and the O_NDELAY flag is set.
- **[EBADMSG]** The message at the *stream head* read queue was not a message containing a passed file descriptor.
- **[EFAULT]** *arg* points outside the allocated address space.
- **[EMFILE]** NOFILES file descriptors are currently open.
- **[ENXIO]** Hangup received on *fdles*.

The following two commands are used for connecting and disconnecting multiplexed STREAMS configurations.

**I_LINK**

Connects two *streams*, where *fdles* is the file descriptor of the *stream* connected to the multiplexing driver, and *arg* is the file descriptor of the *stream* connected to another driver. The *stream* designated by *arg* gets connected below the multiplexing driver. **I_LINK** requires the multiplexing driver to send an acknowledgement message to the *stream head* regarding the linking operation. This call returns a multiplexor ID number (an identifier used to disconnect the multiplexor, see **I_UNLINK**) on success, and a -1 on failure. On failure, *errno* is set to one of the following values:

- **[ENXIO]** Hangup received on *fdles*.
- **[ETIME]** Time out before acknowledgement message was received at *stream head*.
- **[EAGAIN]** Temporarily unable to allocate storage to perform the **I_LINK**.
- **[ENOSR]** Unable to allocate storage to perform the **I_LINK** due to insufficient STREAMS memory resources.
- **[EBADF]** *arg* is not a valid, open file descriptor.
- **[EINVAL]** *fdles stream* does not support multiplexing.
- **[EINVAL]** *arg* is not a *stream*, or is already linked under a multiplexor.
The specified link operation would cause a "cycle" in the resulting configuration; that is, if a given stream head is linked into a multiplexing configuration in more than one place.

An I_LINK can also fail while waiting for the multiplexing driver to acknowledge the link request, if a message indicating an error or a hangup is received at the stream head of fildes. In addition, an error code can be returned in the positive or negative acknowledgement message. For these cases, I_LINK will fail with errno set to the value in the message.

I_UNLINK

Disconnects the two streams specified by fildes and arg. fildes is the file descriptor of the stream connected to the multiplexing driver. fildes must correspond to the stream on which the ioctl I_LINK command was issued to link the stream below the multiplexing driver. arg is the multiplexer ID number that was returned by the I_LINK. If arg is -1, then all Streams which were linked to fildes are disconnected. As in I_LINK, this command requires the multiplexing driver to acknowledge the unlink. On failure, errno is set to one of the following values:

[ENXIO] Hangup received on fildes.

[ETIME] Time out before acknowledgement message was received at stream head.

[ENOSR] Unable to allocate storage to perform the I_UNLINK due to insufficient STREAMS memory resources.

[EINVAL] arg is an invalid multiplexor ID number or fildes is not the stream on which the I_LINK that returned arg was performed.

An I_UNLINK can also fail while waiting for the multiplexing driver to acknowledge the link request, if a message indicating an error or a hangup is received at the stream head of fildes. In addition, an error code can be returned in the positive or negative acknowledgement message. For these cases, I_UNLINK will fail with errno set to the value in the message.
SEE ALSO
   close(2), fcntl(2), intro(2), ioctl(2), open(2), read(2), getmsg(2), poll(2),
   putmsg(2), signal(2), sigset(2), write(2) in the *Programmer's Reference
   Manual.*
   STREAMS Programmer's Guide.
   STREAMS Primer.

DIAGNOSTICS
   Unless specified otherwise above, the return value from ioctl is 0 upon suc-
   cess and -1 upon failure with errno set as indicated.
NAME
t3270 – Silicon Graphics 3270 interface card

DESCRIPTION
This is the driver for the Silicon Graphics 3270 interface card. The card is a communication adapter for the connection of a Silicon Graphics 4D system to an IBM 3270 Cluster Controller using the 3270 Coax Type A protocol.

For Control Unit Terminal (CUT) operations, the coax emulation code is soft-loaded by the driver onto the card to maintain the session, and to manage the contents of the display buffer.

For Distributed Function Terminal (DFT) operations, the coax emulation code is to maintain the communication session, while the driver is responsible for the decoding and execution of the SNA data stream.

The interface between the coax emulation code and the driver is maintained as a dual-ported communication memory area, while the interface between the driver and the application level emulator (t3270(1)) is through ioctl.

```c
#include <sys/t3270reg.h>
di_info_t diaginfo;

ioctl(fd,UCODE_DLOAD,addr);    /* microcode download */
iocll(fd,START_8344,0);        /* start the DP8344 */
iocll(fd,EM_CTRL,ctrl_word);    /* set emulation control */
iocll(fd,SET_DID,id);          /* set device id */
iocll(fd,START_COMM,0);        /* start communication */
iocll(fd,STOP_COMM,0);         /* stop communication */
iocll(fd,CUR_UD,cursor);       /* get cursor */
iocll(fd,SCR_UD,scr);          /* update screen */
iocll(fd,SEND_KEY,key);        /* send key code */
iocll(fd,BEGIN_MSG,0);         /* begin messaging mode */
iocll(fd,STOP_MSG,0);          /* stop messaging mode */
iocll(fd,SEND_MSG,msg_count);  /* send message */
iocll(fd,RECV_MSG,addr);       /* receive message */
iocll(fd,SET_TIME,t);          /* set timeout */
iocll(fd,SELF_TST,&diaginfo);  /* self test */
iocll(fd,REG_TST,&diaginfo);   /* register test */
iocll(fd,DP_MEM_TST,&diaginfo);/* data memory test */
iocll(fd,IT_MEM_TST,&diaginfo);/* instruction memory test */
```

typedef struct {
    u_int pattn_w;          /* pattern written */
    u_int pattn_r;          /* pattern read */
    u_short err_addr;       /* where error occur */
    u_char err_flag;        /* type of error */
};

April 1990 - 1 - Version 5.0


Only one device file may be opened at a time.

FILES
/dev/t3270

SEE ALSO
 t3270(1)
NAME
tcp – Internet Transmission Control Protocol

SYNOPSIS
#include <sys/socket.h>
#include <netinet/in.h>

s = socket(AF_INET, SOCK_STREAM, 0);

DESCRIPTION
The TCP protocol provides reliable, flow-controlled, two-way transmission of data. It is a byte-stream protocol used to support the SOCK_STREAM abstraction. TCP uses the standard Internet address format and, in addition, provides a per-host collection of "port addresses". Thus, each address is composed of an Internet address specifying the host and network, with a specific TCP port on the host identifying the peer entity.

Sockets utilizing the tcp protocol are either "active" or "passive". Active sockets initiate connections to passive sockets. By default TCP sockets are created active; to create a passive socket the listen(2) system call must be used after binding the socket with the bind(2) system call. Only passive sockets may use the accept(2) call to accept incoming connections. Only active sockets may use the connect(2) call to initiate connections.

Passive sockets may "underspecify" their location to match incoming connection requests from multiple networks. This technique, termed "wildcard addressing", allows a single server to provide service to clients on multiple networks. To create a socket which listens on all networks, the Internet address INADDR_ANY must be bound. The TCP port may still be specified at this time; if the port is left unspecified by setting it to 0, the system will assign one. Once a connection has been established the socket’s address is fixed by the peer entity’s location. The address assigned the socket is the address associated with the network interface through which packets are being transmitted and received. Normally this address corresponds to the peer entity’s network.

TCP supports one socket option which is tested with getsockopt(2). Under most circumstances, TCP sends data when it is presented; when outstanding data has not yet been acknowledged, it gathers small amounts of output to be sent in a single packet once an acknowledgement is received. For a small number of clients, such as window systems that send a stream of mouse events which receive no replies, this packetization may cause significant delays. Therefore, TCP provides a boolean option, TCP_NODELAY (from <netinet/tcp.h>), to defeat this algorithm.
Options at the IP transport level may be used with TCP; see `ip(7P)`. Incoming connection requests that are source-routed are noted, and the reverse source route is used in responding.

**DIAGNOSTICS**
A socket operation may fail with one of the following errors returned:

- `[EISCONN]` when trying to establish a connection on a socket which already has one;
- `[ENOBUFFS]` when the system runs out of memory for an internal data structure;
- `[ETIMEDOUT]` when a connection was dropped due to excessive retransmissions;
- `[ECONNRESET]` when the remote peer forces the connection to be closed;
- `[ECONNREFUSED]` when the remote peer actively refuses connection establishment (usually because no process is listening to the port);
- `[EADDRINUSE]` when an attempt is made to create a socket with a port which has already been allocated;
- `[EADDRNOTAVAIL]` when an attempt is made to create a socket with a network address for which no network interface exists.

**SEE ALSO**
`getsockopt(2)`, `socket(2)`, `intro(3)`, `inet(7F)`, `ip(7P)`
NAME
termio, termios – general System V and POSIX terminal interfaces

DESCRIPTION
All of the asynchronous communications ports use the same general interface, no matter what hardware is involved. The remainder of this section discusses the common features of this interface. The POSIX termios interface is a set of library routines built on top of the System V ioctl interface, and is described at the bottom of this man page.

When a terminal file is opened, it normally causes the process to wait until a connection is established. In practice, users’ programs seldom open terminal files; they are opened by getty and become a user’s standard input, output, and error files. The very first time that a process group leader opens a terminal file not already associated with a process group, that terminal file becomes the control terminal for that process group. The control terminal plays a special role in handling quit and interrupt signals, as discussed below. The control terminal is inherited by a child process during a fork(2). A process can break this association by changing its process group using setpgid(2).

When the carrier signal from the data-set drops, a hang-up signal is sent to all processes that have this terminal as the control terminal. Unless other arrangements have been made, this signal causes the processes to terminate. If the hang-up signal is ignored, any subsequent read(2) returns with an end-of-file indication. Thus, programs that read a terminal and test for end-of-file can terminate appropriately when hung up on.

A terminal associated with one of these files ordinarily operates in full-duplex mode. Characters may be typed at any time, even while output is occurring, and are only lost when the system’s character input buffers become completely full, which is rare, or when the user has accumulated the maximum allowed number of input characters that have not yet been read by some program. Currently, this limit is 256 characters. When the input limit is reached, the buffer is flushed and all the saved characters are thrown away without notice.

Normally, terminal input is processed in units of lines. A line is delimited by a new-line (ASCII LF) character, an end-of-file (ASCII EOT) character, or an end-of-line character. This means that a program attempting to read will be suspended until an entire line has been typed. Also, no matter how many characters are requested in the read call, at most one line will be returned. It is not, however, necessary to read a whole line at once; any number of characters may be requested in a read, even one, without losing information.
During input, erase and kill processing is normally done. The erase character erases the last character typed, except that it will not erase beyond the beginning of the line. The kill character kills (deletes) the entire input line, and optionally outputs a new-line character. Both these characters operate on a key-stroke basis, independently of any backspacing or tabbing that may have been done. The erase and kill characters may be changed.

When one or more characters are written, they are transmitted to the terminal as soon as previously-written characters have finished typing. Input characters are echoed by putting them in the output queue as they arrive. If a process produces characters more rapidly than they can be typed, it will be suspended when its output queue exceeds some limit. When the queue has drained down to some threshold, the program is resumed.

Several ioctl(2) system calls apply to terminal files. The primary calls use the following structure, defined in <termio.h>:

```c
typedef unsigned short tcflag_t;
tcflag_t c_iflag; /* input modes */
tcflag_t c_oflag; /* output modes */
tcflag_t c_cflag; /* control modes */
tcflag_t c_lflag; /* local modes */
struct termio {
  char c_line; /* line discipline */
  char c_cc[NCC+NCC_PAD+NCC_EXT]; /* control chars */
};
```

The c_line field defines the line discipline used to interpret control characters. A line discipline is associated with a family of interpretations. For example, LDISC0 is the standard System V set of interpretations, while LDISC1 is similar to the interpretations used in the 4.3BSD `new' tty driver. In LDISC1,

- additional control characters are available (see below),
- control characters which are not editing characters are echoed as `\'' followed by the equivalent letter,
- backspacing does not back up into the prompt,
• input is re-typed when backspacing encounters a confusion between what the user and the computer have typed, and

• job control is available when using *csh*(1).

Certain characters have special functions on input. These functions and their default character values are summarized as follows:

**INTR** (Typically, rubout or ASCII DEL) generates an *interrupt* signal which is sent to all processes with the associated control terminal. Normally, each such process is forced to terminate, but arrangements may be made either to ignore the signal or to receive a trap to an agreed-upon location; see *signal*(2).

**QUIT** (Typically, control-v or ASCII FS) generates a *quit* signal. Its treatment is identical to the interrupt signal except that, unless a receiving process has made other arrangements, it will not only be terminated but a core image file (called *core*) will be created in the current working directory.

**ERASE** (Typically, control-H or backspace) erases the preceding character. It will not erase beyond the start of a line, as delimited by a NL, EOF, or EOL character.

**KILL** (Typically, control-U) deletes the entire line, as delimited by a NL, EOF, or EOL character.

**EOF** (Typically, control-D or ASCII EOT) may be used to generate an end-of-file from a terminal. When received, all the characters waiting to be read are immediately passed to the program, without waiting for a new-line, and the EOF is discarded. Thus, if there are no characters waiting, which is to say the EOF occurred at the beginning of a line, zero characters will be passed back, which is the standard end-of-file indication.

**NL** (ASCII LF) is the normal line delimiter. It can not be changed or escaped.

**EOL** (Typically, ASCII NUL) is an additional line delimiter, like NL. It is not normally used.

**EOL2** is another additional line delimiter.

**STOP** (Typically, control-S or ASCII DC3) can be used to temporarily suspend output. It is useful with CRT terminals to prevent output from disappearing before it can be read. While output is suspended, STOP characters are ignored and not read.
START (Typically, control-Q or ASCII DC1) is used to resume output which has been suspended by a STOP character. While output is not suspended, START characters are ignored and not read. The start/stop characters can not be changed or escaped in LDISC0.

SWITCH
(Control-Z or ASCII SUB) is used by the csh(1) job control facility. Typing this character will cause the foreground job to be suspended.

MIN Used to control I/O during raw mode (ICANON off) processing [see the MIN/TIME Interaction section below.

TIME Used to control I/O during raw mode (ICANON off) processing [see the MIN/TIME Interaction section below.

The following characters have special functions on input when the line discipline is set to LDISC1. These functions and their default character values are summarized as follows:

LNEXT (Control-V) causes the next character input to treated literally.

WERASE (Control-W) erases the preceding white space-delimited word. It will not erase beyond the start of a line, as delimited by a NL, EOF, or EOL character.

RPRNT (Control-R) causes the line to be reprinted.

FLUSHO (Control-O) when typed during output causes the output to be discarded. Typing any character re-enables output.

The character values for INTR, QUIT, ERASE, KILL, EOF, SWITCH, STOP, START, LNEXT, WERASE, RPRNT, FLUSHO, and EOL may be changed to suit individual tastes (see stty(1)). The ERASE, KILL, and EOF characters may be entered literally in LDISC0 by preceding them with the escape character (\), in which case no special function is done and the escape character is not read. In LDISC1, the LNEXT character is used.

The special control characters are defined by the array $c_{cc}$. The relative positions and initial values for each function are as follows:

<table>
<thead>
<tr>
<th>Function</th>
<th>Character</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 VINTR</td>
<td>CINTR</td>
<td>(DEL)</td>
</tr>
</tbody>
</table>
| 1 VQUIT  | CQUIT     | (Control-
| 2 VERASE | CERASE    | (Control-H (backspace)) |
| 3 VKILL  | CKILL     | (Control-U) |
| 4 VEOF   | CEOF      | (Control-D (EOT)) |
| 4 VMIN   |           |       |
| 5 VEOL   | NUL       |       |
| 5 VTIME  |           |       |
| 6 VEOL2  | NUL       |       |

April 1990

- 4 -

Version 5.0
If the line discipline (c_line) is set to LDISC1, then additional control characters are defined:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VLNEXT</td>
<td>(Control-V)/* take next char literally */</td>
</tr>
<tr>
<td>CVNEXT</td>
<td>(Control-W)/* erase previous word */</td>
</tr>
<tr>
<td>VRPRNT</td>
<td>(Control-R)/* retype the line */</td>
</tr>
<tr>
<td>VFLUSHO</td>
<td>(Control-O)/* flush output */</td>
</tr>
<tr>
<td>VSTOP</td>
<td>(Control-S)/* XOFF */</td>
</tr>
<tr>
<td>VSTART</td>
<td>(Control-Q)/* XON */</td>
</tr>
</tbody>
</table>

### Input Modes

The c_iflag field describes the basic terminal input control:

- **IGNBRK**: 0000001 Ignore break condition.
- **BRKINT**: 0000002 Signal interrupt on break.
- **IGNPAR**: 0000004 Ignore characters with parity errors.
- **PARMRK**: 0000010 Mark parity errors.
- **INPCK**: 0000020 Enable input parity check.
- **ISTRIP**: 0000040 Strip character.
- **INLCR**: 0000100 Map NL to CR on input.
- **IGNCR**: 0000200 Ignore CR.
- **ICRNL**: 0000400 Map CR to NL on input.
- **IUCLC**: 0001000 Map upper-case to lower-case on input.
- **IXON**: 0002000 Enable start/stop output control.
- **IXANY**: 0004000 Enable any character to restart output.
- **IXOFF**: 0010000 Enable start/stop input control.
- **IBLKMD**: 0020000 Enable

If IGNBRK is set, the break condition (a character framing error with data all zeros) is ignored, that is, not put on the input queue and therefore not read by any process. Otherwise if BRKINT is set, the break condition will generate an interrupt signal and flush both the input and output queues. If IGNPAR is set, characters with other framing and parity errors are ignored.

If PARMRK is set, a character with a framing or parity error which is not ignored is read as the three-character sequence: 0377, 0, X, where X is the data of the character received in error. To avoid ambiguity in this case, if ISTRIP is not set, a valid character of 0377 is read as 0377, 0377. If PARMRK is not set, a framing or parity error which is not ignored is read as the character NUL (0).
If **INPCK** is set, input parity checking is enabled. If **INPCK** is not set, input parity checking is disabled. This allows output parity generation without input parity errors.

If **ISTRIP** is set, valid input characters are first stripped to 7-bits, otherwise all 8-bits are processed.

If **INLCR** is set, a received NL character is translated into a CR character. If **IGNCR** is set, a received CR character is ignored (not read). Otherwise if **ICRNL** is set, a received CR character is translated into a NL character.

If **IUCLC** is set, a received upper-case alphabetic character is translated into the corresponding lower-case character.

If **IXON** is set, start/stop output control is enabled. A received STOP character will suspend output and a received START character will restart output. All start/stop characters are ignored and not read.

If **IXANY** is set, any input character will restart output which has been suspended.

If **IXOFF** is set, the system will transmit START/STOP characters when the input queue is nearly empty/full.

The initial input control value is all-bits-clear.

### Output Modes

The **c_oflag** field specifies the system treatment of output:

- **OPOST** 0000001 Postprocess output.
- **OLCUC** 0000002 Map lower case to upper on output.
- **ONLCR** 0000004 Map NL to CR-NL on output.
- **OCRNL** 0000010 Map CR to NL on output.
- **ONOCR** 0000020 No CR output at column 0.
- **ONLRET** 0000040 NL performs CR function.
- **OFILL** 0000100 Use fill characters for delay.
- **OFDEL** 0000200 Fill is DEL, else NUL.
- **NLDLY** 0000400 Select new-line delays:
  - **NL0** 0
  - **NL1** 0000400
- **CRDLY** 0003000 Select carriage-return delays:
  - **CR0** 0
  - **CR1** 0001000
  - **CR2** 0002000
  - **CR3** 0003000
- **TABDLY** 0014000 Select horizontal-tab delays:
  - **TAB0** 0
  - **TAB1** 0004000
  - **TAB2** 0010000
<table>
<thead>
<tr>
<th>Code</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TAB3</td>
<td>0014000</td>
<td>Expand tabs to spaces.</td>
</tr>
<tr>
<td>BSDLY</td>
<td>0020000</td>
<td>Select backspace delays:</td>
</tr>
<tr>
<td>BS0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>BS1</td>
<td>0020000</td>
<td></td>
</tr>
<tr>
<td>VTDLY</td>
<td>0040000</td>
<td>Select vertical-tab delays:</td>
</tr>
<tr>
<td>VT0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>VT1</td>
<td>0040000</td>
<td></td>
</tr>
<tr>
<td>FFDLY</td>
<td>0100000</td>
<td>Select form-feed delays:</td>
</tr>
<tr>
<td>FF0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>FF1</td>
<td>0100000</td>
<td></td>
</tr>
</tbody>
</table>

If **OPOST** is set, output characters are post-processed as indicated by the remaining flags, otherwise characters are transmitted without change.

If **OLCUC** is set, a lower-case alphabetic character is transmitted as the corresponding upper-case character. This function is often used in conjunction with **IUCLC**.

If **ONLCR** is set, the **NL** character is transmitted as the **CR-NL** character pair. If **OCRNL** is set, the **CR** character is transmitted as the **NL** character. If **ONOCR** is set, no **CR** character is transmitted when at column 0 (first position). If **ONLRET** is set, the **NL** character is assumed to do the carriage-return function; the column pointer will be set to 0 and the delays specified for **CR** will be used. Otherwise the **NL** character is assumed to do just the line-feed function; the column pointer will remain unchanged. The column pointer is also set to 0 if the **CR** character is actually transmitted.

The delay bits specify how long transmission stops to allow for mechanical or other movement when certain characters are sent to the terminal. In all cases a value of 0 indicates no delay. If **OFILL** is set, fill characters will be transmitted for delay instead of a timed delay. This is useful for high baud rate terminals which need only a minimal delay. If **OFDEL** is set, the fill character is **DEL**, otherwise **NUL**.

If a form-feed or vertical-tab delay is specified, it lasts for about 2 seconds.

New-line delay lasts about 0.10 seconds. If **ONLRET** is set, the carriage-return delays are used instead of the new-line delays. If **OFILL** is set, two fill characters will be transmitted.

Carriage-return delay type 1 is dependent on the current column position, type 2 is about 0.10 seconds, and type 3 is about 0.15 seconds. If **OFILL** is set, delay type 1 transmits two fill characters, and type 2, four fill characters.
Horizontal-tab delay type 1 is dependent on the current column position. Type 2 is about 0.10 seconds. Type 3 specifies that tabs are to be expanded into spaces. If OFILL is set, two fill characters will be transmitted for any delay.

Backspace delay lasts about 0.05 seconds. If OFILL is set, one fill character will be transmitted.

The actual delays depend on line speed and system load.

The initial output control value is all bits clear.

**Control Modes**

The `c_flag` field describes the hardware control of the terminal:

<table>
<thead>
<tr>
<th>Code</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CBAUD</td>
<td>0000017</td>
<td>Baud rate:</td>
</tr>
<tr>
<td>B0</td>
<td>0</td>
<td>Hang up</td>
</tr>
<tr>
<td>B50</td>
<td>0000001</td>
<td>50 baud</td>
</tr>
<tr>
<td>B75</td>
<td>0000002</td>
<td>75 baud</td>
</tr>
<tr>
<td>B110</td>
<td>0000003</td>
<td>110 baud</td>
</tr>
<tr>
<td>B134</td>
<td>0000004</td>
<td>134 baud</td>
</tr>
<tr>
<td>B150</td>
<td>0000005</td>
<td>150 baud</td>
</tr>
<tr>
<td>B200</td>
<td>0000006</td>
<td>200 baud</td>
</tr>
<tr>
<td>B300</td>
<td>0000007</td>
<td>300 baud</td>
</tr>
<tr>
<td>B600</td>
<td>0000010</td>
<td>600 baud</td>
</tr>
<tr>
<td>B1200</td>
<td>0000011</td>
<td>1200 baud</td>
</tr>
<tr>
<td>B1800</td>
<td>0000012</td>
<td>1800 baud</td>
</tr>
<tr>
<td>B2400</td>
<td>0000013</td>
<td>2400 baud</td>
</tr>
<tr>
<td>B4800</td>
<td>0000014</td>
<td>4800 baud</td>
</tr>
<tr>
<td>B9600</td>
<td>0000015</td>
<td>9600 baud</td>
</tr>
<tr>
<td>B19200</td>
<td>0000016</td>
<td>19200 baud</td>
</tr>
<tr>
<td>EXTA</td>
<td>0000016</td>
<td>External A</td>
</tr>
<tr>
<td>B38400</td>
<td>0000017</td>
<td>38400 baud</td>
</tr>
<tr>
<td>EXTB</td>
<td>0000017</td>
<td>External B</td>
</tr>
<tr>
<td>CSIZE</td>
<td>0000060</td>
<td>Character size:</td>
</tr>
<tr>
<td>CS5</td>
<td>0</td>
<td>5 bits</td>
</tr>
<tr>
<td>CS6</td>
<td>0000020</td>
<td>6 bits</td>
</tr>
<tr>
<td>CS7</td>
<td>0000040</td>
<td>7 bits</td>
</tr>
<tr>
<td>CS8</td>
<td>0000060</td>
<td>8 bits</td>
</tr>
<tr>
<td>CSTOPB</td>
<td>0000100</td>
<td>Send two stop bits, else one.</td>
</tr>
<tr>
<td>CREAD</td>
<td>0000200</td>
<td>Enable receiver.</td>
</tr>
<tr>
<td>PARENB</td>
<td>0000400</td>
<td>Parity enable.</td>
</tr>
<tr>
<td>PARODD</td>
<td>0001000</td>
<td>Odd parity, else even.</td>
</tr>
<tr>
<td>HUPCL</td>
<td>0002000</td>
<td>Hang up on last close.</td>
</tr>
<tr>
<td>CLOCAL</td>
<td>0004000</td>
<td>Local line, else dial-up.</td>
</tr>
<tr>
<td>RCV1EN</td>
<td>0010000</td>
<td></td>
</tr>
</tbody>
</table>
XMT1EN  0020000
LOBLK   0040000  Block layer output.

SSPEED   B9600  Default baud rate.

The CBAUD bits specify the baud rate. The zero baud rate, B0, is used to hang up the connection. If B0 is specified, the data-terminal-ready signal will not be asserted. Normally, this will disconnect the line. For any particular hardware, impossible speed changes are ignored.

The CSIZE bits specify the character size in bits for both transmission and reception. This size does not include the parity bit, if any. If CSTOPB is set, two stop bits are used, otherwise one stop bit. For example, at 110 baud, two stops bits are required.

If PARENB is set, parity generation and detection is enabled and a parity bit is added to each character. If parity is enabled, the PARODD flag specifies odd parity if set, otherwise even parity is used.

If CREAD is set, the receiver is enabled. Otherwise no characters will be received.

If HUPCL is set, the line will be disconnected when the last process with the line open closes it or terminates. That is, the data-terminal-ready signal will not be asserted.

If CLOCAL is set, the line is assumed to be a local, direct connection with no modem control. Otherwise modem control is assumed.

If LOBLK is set, the output of a job control layer will be blocked when it is not the current layer. Otherwise the output generated by that layer will be multiplexed onto the current layer.

The initial hardware control value after open usually is B9600, CS8, CREAD, HUPCL. These initial values depend on the driver; see duart(7) and csdio(7).

The c_iflag field of the argument structure is used by the line discipline to control terminal functions. The basic line discipline (LDISCO) provides the following:

ISIG      0000001  Enable signals.
ICANON    0000002  Canonical input (erase and kill processing).
XCASE     0000004  Canonical upper/lower presentation.
ECHO      0000010  Enable echo.
ECHOE     0000020  Echo erase character as BS-SP-BS.
ECHOK     0000040  Echo NL after kill character.
ECHONL    0000100  Echo NL.
NOFLSH    0000200  Disable flush after interrupt or quit.
IIEXTEN   0000400  Enable extended functions (not yet implemented)
ITOSTOP  0001000  Send SIGHTOU for background output.

If ISIG is set, each input character is checked against the special control
characters INTR, SWITCH, and QUIT. If an input character matches one
of these control characters, the function associated with that character is
performed. If ISIG is not set, no checking is done. Thus these special
input functions are possible only if ISIG is set. These functions may be dis-
abled individually by changing the value of the control character to an
unlikely or impossible value (e.g., 0377).

If ICANON is set, canonical processing is enabled. This enables the erase
and kill edit functions, and the assembly of input characters into lines del-
imited by NL, EOF, and EOL. If ICANON is not set, read requests are
satisfied directly from the input queue. A read will not be satisfied until at
least MIN characters have been received or the timeout value TIME has
expired between characters. This allows fast bursts of input to be read
efficiently while still allowing single character input. The MIN and TIME
values are stored in the position for the EOF and EOL characters, respec-
tively. The time value represents tenths of seconds. See the section
MIN/TIME Interaction below.

If XCASE is set, and if ICANON is set, an upper-case letter is accepted on
input by preceding it with a \ character, and is output preceded by a \ char-
acter. In this mode, the following escape sequences are generated on output
and accepted on input:

\n  \n  |  !
  -  '
  {  ()
  }  )
\  \n
For example, A is input as \a, \n as \n, and \N as \n.

If ECHO is set, characters are echoed as received.

When ICANON is set, the following echo functions are possible. If ECHO
and ECHOE are set, the erase character is echoed as ASCII BS SP BS,
which will clear the last character from a CRT screen. If ECHOE is set
and ECHO is not set, the erase character is echoed as ASCII SP BS. If
ECHOK is set, the NL character will be echoed after the kill character to
emphasize that the line will be deleted. Note that an escape character
preceding the erase or kill character removes any special function. If
ECHONL is set, the NL character will be echoed even if ECHO is not set.
This is useful for terminals set to local echo (so-called half duplex). Unless
escaped, the EOF character is not echoed. Because EOT is the default
EOF character, this prevents terminals that respond to EOT from hanging up.

If NOFLSH is set, the normal flush of the input and output queues associated with the quit, switch, and interrupt characters will not be done.

The initial line-discipline control value is all bits clear.

MIN/TIME Interaction

MIN represents the minimum number of characters that should be received when the read is satisfied (i.e., the characters are returned to the user). TIME is a timer of 0.10 second granularity used to time-out bursty and short-term data transmissions. The four possible values for MIN and TIME and their interactions follow:

1. \( \text{MIN} > 0 \), \( \text{TIME} > 0 \) In this case, TIME serves as an inter-character timer activated after the first character is received, and reset upon receipt of each character. MIN and TIME interact as follows:
   
   As soon as one character is received the inter-character timer is started.
   
   If MIN characters are received before the inter-character timer expires the read is satisfied.
   
   If the timer expires before MIN characters are received the characters received to that point are returned to the user.

   A \textit{read}(2) operation will sleep until the MIN and TIME mechanisms are activated by the receipt of the first character; thus, at least one character must be returned.

2. \( \text{MIN} > 0 \), \( \text{TIME} = 0 \). In this case, because \( \text{TIME} = 0 \), the timer plays no role and only MIN is significant. A read operation is not satisfied until MIN characters are received.

3. \( \text{MIN} = 0 \), \( \text{TIME} > 0 \) In this case, because \( \text{MIN} = 0 \), TIME no longer serves as an inter-character timer, but now serves as a read timer that is activated as soon as the read operation is processed (in canon). A read operation is satisfied as soon as a single character is received or the timer expires, in which case the read operation will not return any characters.

4. \( \text{MIN} = 0 \), \( \text{TIME} = 0 \) In this case, return is immediate. If characters are present, they will be returned to the user.
System Calls

The primary `ioctl(2)` system calls have the form:

```c
ioctl (filedes, command, arg)
struct termio *arg;
```

The commands using this form are:

- **TCGETA** Get the parameters associated with the terminal and store in the `termio` structure referenced by `arg`.
- **TCSETA** Set the parameters associated with the terminal from the structure referenced by `arg`. The change is immediate.
- **TCSETAW** Wait for the output to drain before setting the new parameters. This form should be used when changing parameters that will affect output.
- **TCSETAF** Wait for the output to drain, then flush the input queue and set the new parameters.

Additional `ioctl(2)` calls have the form:

```c
ioctl (filedes, command, arg)
int arg;
```

The commands using this form are:

- **TCSBRK** Wait for the output to drain. If `arg` is 0, then send a break (zero bits for 0.25 seconds).
- **TCXONC** Start/stop control. If `arg` is 0, suspend output; if 1, restart suspended output.
- **TCFLSH** If `arg` is 0, flush the input queue; if 1, flush the output queue; if 2, flush both the input and output queues.

The following `ioctl` calls apply only to pseudo terminals; see `pty(7M)` for their descriptions:

```
TIOCPKT, TIOCPKT_DATA, TIOCPKT_FLUSHREAD,
TIOCPKT_FLUSHWRITE, TIOCPKT_STOP, TIOCPKT_START,
TIOCPKT_NOSTOP, TIOCPKT_DOSTOP
```

**TIOCNOTTY**

Disconnect calling process from terminal and process group. NOTE: IRIX, unlike BSD, sets the process’s process group ID to its process ID. Berkeley’s `TIOCNOTTY` sets the process group to zero.
TIOCSTI
Simulate terminal input: arg points to a character which the system pretends had been typed on the terminal.

TIOCSPGRP
Set process group of tty: arg is a pointer to an int which is the value to which the process group ID for this terminal will be set.

TIOCGPGRP
Get process group of tty: arg is a pointer to an int into which is placed the process group ID of the process group for which this terminal is the control terminal.

TIOCFflush
If the int pointed to by arg has a zero value, all characters waiting in input or output queues are flushed. Otherwise, the value of the int is for the FREAD and FWRITE bits defined in <sys/file.h>; if the FREAD bit is set, all characters waiting in input queues are flushed, and if the FWRITE bit is set, all characters waiting in output queues are flushed.

Window size structure:

```c
struct winsize {
    unsigned short ws_row, ws_col; /* size in chars */
    unsigned short ws_xpixel, ws_ypixel; /* size in pixels */
};
```

TIOCGWINSZ
Get window size.

TIOCSWINSZ
Set window size.

POSIX TERMIOS

```
#define VSUSP VSWITCH
#define TCSANOW 0 /* change term attributes NOW */
#define TCSADRAIN 1 /* change after current output written */
#define TCSAFLUSH 2 /* as DRAIN, but toss pending input */

/* For tcflush sys call, designate which queue to flush */
#define TCIFLUSH 0
```
#define TCOFLUSH 1
#define TCIOFLUSH 2

/* For tcflow sys call, below defines specify on/off & input/output */
#define TCOFF 0
#define TCOON 1
#define TCIOFF 2
#define TCION 3

typedef uint speed_t;

#define TO_STOP LOBLK /* send SIGTTOU for background output */

/*
 * POSIX termios control structure.
 * Note that this structure is identical with "struct termio"
 * defined in <sys/termio.h>.
 */

struct termios {
    tcflag_t c_iflag; /* input modes */
    tcflag_t c_oflag; /* output modes */
    tcflag_t c_cflag; /* control modes */
    tcflag_t c_lflag; /* line discipline modes */
    char c_line; /* line disc. selector */
    cc_t  c_cc[NCCS]; /* control chars */
};

/* Two extra POSIX iflags -- other iflags are in <sys/termio.h> */
#define IEXTEN IIXTEN /* NOT YET IMPLEMENTED */
#define TOSTOP ITOSTOP /* NOT YET IMPLEMENTED */

NOTES

Of the ioctl commands listed above, all except TCGETA alter the state of
the terminal. For this reason, a background job which issues any of com-
mands except TCGETA will be suspended. Refer to csh(1) for more infor-
mation about job control.

FILES
/dev/tty*

SEE ALSO

fork(2), ioctl(2), setpgid(2), and signal(2) in the Programmer's Reference
Manual.
pty(7M) in the *Administrator’s Reference Manual*.

Extensions by Silicon Graphics, Inc.
NAME
tps – SCSI 1/4-inch Cartridge tape interface

SYNOPSIS
/dev/mt/tps*
/dev/rmt/tps*

DESCRIPTION
The IRIS-4D series workstations support the Small Computer System Interface (SCSI) for various tape drives, including QIC24 and QIC150 1/4" cartridges, 9 track tapes, and 8 mm video tapes. The 1/4" cartridges are compatible with the IRIS 2000 and 3000 series workstations (see NOTE below).

The special files are named according to the convention discussed in intro(7):

/dev/{r}mt/tps<controller-#/d<ID-#/>{nr}<{ns}v}{density}

Where {nr} is the no-rewind on close device, {ns} is the non-byte swapping device, and {v} is the variable block size device (supported only for 9 track and 8 mm), and (for 9 track drives only) density is one of 800, 1600, 3200, or 6250.

These special devices are accessible by only one user or program at a time.

Typically, if this device is used as the system tape drive, the device specific names described here are linked to user friendly names in the /dev directory. see NOTES below, and mtdio(7) for a description of these names.

ERROR RETURNS
The following errors are returned by this driver:

EAGAIN Unable to allocate memory for the swap buffer when using the byte swapping device; or The drive returned an error indicating it was not ready (tape ejected, drive taken off-line, etc.)

EBUSY Returned on opens when the drive has already been opened.

EFAULT A bad address was passed in a call that required a data transfer.

EINVAL This is returned for requests that are invalid for one reason or another; these include: Attempting to write or write file-mark after reading. Attempting to read after writing. Using a invalid count on read, write, write file-mark, etc. Attempting to do MTAFILE on a drive that isn’t a 9 track drive.
Attempting to do an ioctl on a drive that doesn’t support it (such as MTBSR on Cipher 540S), or attempting to do an unsupported MTOP operation, or other unsupported ioctl’s. Attempting to write to a QIC24 cartridge from a QIC150 drive. Attempting to do something when not at BOT that can only be done at BOT, such as writing or reading a Kennedy tape drive at a different density or speed than was previously used, or switching from the variable block size device to the fixed block size device.

EIO A generic error occurred, such as a SCSI bus reset, unrecoverable media error, etc.

ENOMEM An attempt was made to read data with a count less than that at which the block was written; can only happen with drives that support variable block sizes.

ENOSPC On read or space commands that encounter end of tape or end of data; writes that are attempted at end of tape; also some other commands that encounter EOT or EOD.

EROF S A write or write file-mark was attempted to a write-protected tape.

ENXIO An open was attempted on a device created with an incorrect minor number, or the SCSI ID is in use by some other driver.

NOTE

High density tapes such as the 3M 600 XTD written on an Personal Iris, or other system equipped with high density tape drives can NOT be read on older systems. Even if a "low density" tape (such as 3M 600A) is used, it is still written at a higher density than older tape drives can read. Tapes written on the older systems can still be read on the new tape drives, however. Systems with high density cartridge tape drives such as the Personal Iris are able to read QIC24 tapes (310 oersted) such as the 3M 300XL, but are not able to write them.

8 mm tape drives have /dev/tape linked to the {ns} device for performance, since there is no compatibility issue; 9 track drives are linked to the {ns} {v}.6250 device.

FILES

/dev/mt/tps*
/dev/rmt/tps*

SEE ALSO

bru(1), cpio(1), mt(1), tar(1), ts(7M), tps(7M), xmt(7M)
NAME
ts – ISI VME-QIC2/X cartridge tape controller

SYNOPSIS
/dev/mt/ts*
/dev/rmt/ts*

DESCRIPTION
The IRIS-4D series system supports the ISI quarter inch cartridge tape, using the QIC-02 format, and is compatible with the IRIS series 2000 and IRIS series 3000 workstations. The special files are named according to the convention discussed in intro(7):

/dev/[r]mt/ts<controller-#>d<slave-#>{nr}{ns}{.density}

Currently, only one controller and one slave per controller are supported, so the <controller-#> and <slave-#> are always 0. There are no choices for density, and that portion of the device name is always blank.

These special devices are accessible by only one user at a time.

Typically, if this device is used as the system tape drive, the device specific names described here are linked to user friendly names in the /dev directory. The following table lists the device specific name and its corresponding user friendly name:

<table>
<thead>
<tr>
<th>Device Specific Name</th>
<th>User Friendly Name</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>/dev/mt/ts0d0</td>
<td>/dev/tape</td>
<td>Rewind device (Standard Interface). Bytes are swapped in order to be compatible with the IRIS 2000 and 3000 workstation family.</td>
</tr>
<tr>
<td>/dev/mt/ts0d0nr</td>
<td>/dev/nrtape</td>
<td>Non-rewinding device. Bytes are swapped in order to be compatible with the IRIS 2000 and 3000 workstation family.</td>
</tr>
<tr>
<td>/dev/mt/ts0d0ns</td>
<td>/dev/tapens</td>
<td>Rewind device.</td>
</tr>
</tbody>
</table>
Bytes are not swapped.

/dev/mt/ts0d0nrns  /dev/nrtapens Non-rewinding device. Bytes are not swapped.

FILES
   /dev/mt/ts*
   /dev/rmt/ts*

SEE ALSO
cpio(1), mt(1), tar(1), intro(7), mtio(7)
NAME
tty – controlling terminal interface

DESCRIPTION
The file /dev/tty is, in each process, a synonym for the control terminal associated with the process group of that process, if any. It is useful for programs or shell sequences that wish to be sure of writing messages on the terminal no matter how output has been redirected. It can also be used for programs that demand the name of a file for output, when typed output is desired and it is tiresome to find out what terminal is currently in use.

FILES
/dev/tty
/dev/tty*

SEE ALSO
console(7)
NAME
udp – Internet User Datagram Protocol

SYNOPSIS
#include <sys/socket.h>
#include <netinet/in.h>

s = socket(AF_INET, SOCK_DGRAM, 0);

DESCRIPTION
UDP is a simple, unreliable datagram protocol which is used to support the
SOCK_DGRAM abstraction for the Internet protocol family. UDP sockets
are connectionless, and are normally used with the sendto and recvfrom
calls, though the connect(2) call may also be used to fix the destination for
future packets (in which case the recv(2) or read(2) and send(2) or write(2)
system calls may be used).

UDP address formats are identical to those used by TCP. In particular UDP
provides a port identifier in addition to the normal Internet address format.
Note that the UDP port space is separate from the TCP port space (i.e., a
UDP port may not be "connected" to a TCP port).

When binding a UDP socket, if the local port is unspecified (i.e., set to 0),
the system will choose an appropriate port number for it. In addition broad-
cast packets may be sent (assuming the underlying network supports this)
by using a reserved "broadcast address"; this address is network interface
dependent.

Options at the IP transport level may be used with UDP; see ip(7P).

DIAGNOSTICS
A socket operation may fail with one of the following errors returned:

[EISCONN] when trying to establish a connection on a socket which
already has one, or when trying to send a datagram with
the destination address specified and the socket is already
connected;

[ENOTCONN] when trying to send a datagram, but no destination
address is specified, and the socket hasn't been con-
connected;

[ENOBUFFS] when the system runs out of memory for an internal data
structure;

[EADDRINUSE] when an attempt is made to create a socket with a port
which has already been allocated;
[EADDRNOTAVAIL]
when an attempt is made to create a socket with a network address for which no network interface exists.

SEE ALSO
getssockopt(2), recv(2), send(2), socket(2), intro(3), inet(7F), ip(7P), tcp(7P)
NAME
vh – disk volume header.

DESCRIPTION
The volume header is a small partition at the start of every SGI disk. Typically it occupies about 1 megabyte, exact size depends on disk geometry since it is an integral number of cylinders.
It contains identifying information for the disk, and may also contain some files used in standalone operations when the operating system is not running.
The volume header of the system root disk may be accessed as /dev/vh (or /dev/vrh, for the block and character devices respectively).
The volume header partitions of other disks may be accessed via the appropriate files in the /dev/dsk or /dev/rdsk directories; the volume header special files are identified by the suffix ‘vh’.

The first sector of the volume header (and hence the first sector on the entire disk) contains the disk label. The structure of this is defined in the file <sys/dvh.h>. The file is quite extensively commented, and it is worth referring to it as a supplement to the information in this man page. For security, the label sector is also replicated at the first sector of each track of the first cylinder.
The disk label contains the following sections:

Magic Number
This is a unique number which serves to verify that the sector does contain a label.

Root Partition Identifier.
If the disk is a root disk, this field identifies the partition to be used as the root partition. The information is used by the boot PROMs when the system is booting up.

Swap Partition Identifier.
This is also information used during system boot.

Boot Filename
This is also information used during system boot, it identifies the executable file to be loaded by default (normally the operating system).

Device Parameters.
This section contains information about the particular disk drive (e.g. number of cylinders). It is retrieved by the driver during system startup, and used to configure the disk controller with the parameters it needs to work correctly with the drive.
Volume Directory
As well as the label, the volume header partition can be used to store files in the remaining space. Note that these are NOT regular system files; they are accessed only via the PROM or standalone shell and will not appear in any regular directory. Typically they will be special executable files such as the standalone version of the disk utility fx(1M). They are used for performing special-purpose tasks on the machine when the operating system is not running.

The Volume Directory is a table in the disk label which enables such files to be located by the PROMs. Files in the volume header may be added or removed by dvhtool(1m).

Partitions
Disk space is required for a number of different purposes in a system. It is not usually practical or desirable to dedicate an entire physical disk to each use, so the disk surface is divided into a number of partitions. Typically a partition will contain a filesystem (such as /usr), or will be used for non-filesystem storage, such as swap. The volume header itself is also a partition. Partitions may overlap: for example, the vol partition is defined to be the whole disk, (useful for disk cloning).

In order to direct I/O requests to the correct part of the disk, the driver must know the layout of the partitions. This is specified in the partition table contained in the disk label.

The disk label is set up by fx when the disk is first initialized for the system. A convenient summary of its contents may be printed by prtvtoc(1m). dvhtool gives more detailed information, and is also used for manipulating files in the header partition.

FILES
/dev/vh
/dev/rvh
/dev/dsk/ips#d#vh
/dev/rdsk/ips#d#vh
/dev/dsk/dks#d#vh
/dev/rdsk/dks#d#vh

SEE ALSO
dvhtool(1M), prtvtoc(1M), fx(1m).
NAME
xmt – Xylogics 1/2 inch magnetic tape controller

SYNOPSIS
/dev/mt/xmt*
/dev/rmt/xmt*

DESCRIPTION
This is the driver for the Xylogics Model 772 1/2 inch magnetic tape controller. The driver supports any Cipher 88X or 99X tape drive. Reads and writes of up to 64k bytes are allowed. The special files are named according to the convention discussed in intro(7):

/dev/{r}mt/xmt<controller-#>d<slave-#>{nr}{.density}

Valid densities are one of 800, 1600, 3200, or 6250. Cooked tape files exist in the directory /dev/mt. Raw tape files exist in the directory /dev/rmt. For example, /dev/rmt/xmt0d0nr.1600 is the name of the raw no-rewind 1600 bpi tape device.

Only one of the device files associated with a single drive may be opened at a time. Currently, only the raw tape devices are supported. The cooked tape files point to the same devices as the raw tape files.

IOCTL FUNCTIONS
As well as normal read and write operations, the driver supports a number of special ioctl(2) operations. Commands values and data structures for these are defined in mtio(7).

FILES
/dev/mt/xmt*
/dev/rmt/xmt*

SEE ALSO
cpio(1), mt(1), mtio(7M), tar(1)
NAME
xyl, xyl754 – Xylogics disk controllers and driver.

SYNOPSIS
/dev/dsk/xyl*
/dev/rdsk/xyl*

DESCRIPTION
The IRIS-4D system supports the Xylogics 754 SMD disk controller. A system may contain a maximum of 2 Xylogics controllers, and each controller may have up to 4 SMD disk drives connected to it. The special files are named according to the convention discussed in intro(7M):

/dev/{r}dsk/xyl<controller-#>d<drive-#>{s<partition-#>|vh|vol}

Controller numbers run from 0 to 1 for the two possible controllers in a system. Drive numbers run from 0 to 3.

The kernel driver for the xylogics controller is referred to as xyl754.

IOCTL FUNCTIONS
As well as normal read and write operations, the driver supports a number of special io control (ioctl) operations when opened via the character special file in /dev/rdsk. These may be invoked from a user program by the ioctl system call, see ioctl(2). Command values for these are defined in the system include file <sys/dkio.h>. These ioctl operations are intended for the use of special-purpose disk utilities. Many of them can have drastic or even fatal effects on disk operation if misused; they should be invoked only by the knowledgeable and with extreme caution!

A list of the ioctl commands supported by the xyl754 driver is given below.

DIOCGETVH
reads the disk volume header from the driver into a buffer in the calling program. The arg in the ioctl call is expected to point to a buffer of size at least 512 bytes. The structure of the volume header is defined in the include file <sys/dvh.h>.

DIOCHANDSHAKE
reads identifying data from the controller firmware into a buffer in the calling program. The arg of the ioctl call should point to a buffer at least 16 bytes long. The identifying data is a 'struct CONTROLLER_TABLE', defined in <sys/xl.h>.

April 1990 - 1 - Version 5.0
DIOCRAWREAD
This is something of a misnomer; what it actually does is to invoke a controller command for reading the flawmap information placed on the disk by the original manufacturer in accordance with the SMD specification. The amount of data returned to the calling program is always 512 bytes and the program must provide a buffer of this size to receive it. The operation is controlled by a structure which must be pointed to by the arg of the ioctl call; this is a 'struct rawread_info' which is defined in <sys/dvh.h>. Flawmaps are always at the start of tracks, however for historical reasons the identifying argument is an absolute blocknumber rather than a cylinder/head specification. Thus it is necessary to know the geometry of the disk in order to construct the arguments. For the interpretation of the returned data, see the SMD interface specification.

DIOCSENSE
causes the current controller parameters to be printed on the system console. These are the fields of a 'struct CONTROLLER_TABLE', defined in <sys/xl.h>.

The remaining commands should be used with extreme caution, since misuse can render a disk at least temporarily unreadable, or at worst destroy data on the disk.

DIOCSETVH
transfers a new volume header into the driver, and causes the disk drive to be reconfigured in accordance with the parameters in the new volume header. The arg in the ioctl call is expected to point to a buffer containing a valid volume header with parameters appropriate for the drive.

DIOCFMTMAP
formats a track on the disk, or maps a track to a replacement track, according to information passed in a buffer pointed to by the arg of the ioctl call. The arg should point to a 'struct fmt_map_info', which is defined in the include file <sys/dvh.h>. Note that although this contains definitions for a number of other possible actions, only formatting a track and mapping a track are currently supported. The track(s) to be acted on must be specified in the form of cylinder/head numbers, and these are absolute numbers starting from the beginning of the disk (that is to say, NOT from start of the currently open partition).
FILES
/dev/dsk/xy1*
/dev/rdsk/xy1*
/usr/include/sys/dkio.h /usr/include/sys/dvh.h /usr/include/sys/x1.h

BUGS
For historical reasons, the names of the iocIts are not always logical, and their arguments are an irregular mess.
NAME
zero – source of zeroes

DESCRIPTION
A zero special file is a source of zeroed unnamed memory.
Reads from a zero special file always return a buffer full of zeroes. The file
is of infinite length.
Writes to a zero special file are always successful, but the data written is
ignored.
Mapping a zero special file creates a zero-initialized unnamed memory
object of a length equal to the length of the mapping and rounded up to the
nearest page size as returned by getpagesize(2). Multiple processes can
share such a zero special file object provided a common ancestor mapped
the object MAP_SHARED.

FILES
/dev/zero

SEE ALSO
fork(2), mmap(2)