Altos UNIX® System V/386 Release 3.2

System Administrator's Reference (ADM, HW)

Document History

EDITION	PART NUMBER	DATE
Preliminary Edition	690-23416-001A	February 1990
First Edition	690-23416-001	April 1990
Second Edition	690-23416-002	March 1991

Copyright Notice

Trademarks

The Altos logo, as it appears in this manual, is a registered trademark of Altos Computer Systems.

386 and 486 are trademarks of Intel Corporation.

"ACER Fast File System" is a trademark of ACER Technologies Corporation.

Microsoft, MS-DOS, and XENIX are registered trademarks of Microsoft Corporation.

UNIX is a registered trademark of UNIX System Laboratories, Inc.

Limitations

Altos Computer Systems reserves the right to make changes to the product described in this manual at any time and without notice. Neither Altos nor its suppliers make any warranty with respect to the accuracy of the information in this manual.

GUIDE TO YOUR ALTOS UNIX® SYSTEM V/386 RELEASE 3.2 DOCUMENTATION

RUN-TIME SYSTEM

These books come with every system:

Installation Guide
Part Number: 690–24096–*nnn*
- Operating System installation
  Upgrade procedure

System Administrator's Guide
Part Number: 690–23415–*nnn*
- Sysadmsh
- Security
- System tuning, troubleshooting
- Peripherals
- Virtual Disks

User's Guide
Part Number: 690–23408–*nnn*
- Vi, ed, mail, awk, sed
- Shells: sh and csh
- Job scheduling commands

User's Reference (C, M, F)
Part Number: 690–23414–*nnn*
(also provided online with each operating system)
- (C) Commands
- (M) Miscellaneous files and commands
- (F) File formats

System Administrator's Reference (ADM, HW)
Part Number: 690–23416–*nnn*
(also provided online with each operating system)
- (ADM) Administrative commands
- (HW) Hardware information

These books may be ordered separately:

Using the AOM Menu System
Part Numbers: 690–23814–*nnn*
- Easy–to–use menus to use programs
- Menu manager to add, update, remove menus

Tutorial
Part Number: 690–23407–*nnn*
- Basic concepts and tasks
  Files and directories
- Utilities

International Operating System Guide
Part Number: 690–23810–*nnn*
- Character sets
- 7–bit vs. 8–bit characters

DEVELOPMENT SYSTEM

Set Part Number: 690–23417–000

Programmer's Reference (CP,S)
- (CP) Programming commands
- (S) System services, library routines

Programmer's Guide
- Lex, lint, yacc
- SCCS, make
- Extended Terminal Interface (ETI)
- Sdb, adb
- Shared libraries
- File and record locking

C Language Guide
- C User's Guide
- C Language Reference

Library Guide
- C Library Guide
- XENIX Development and Portability Guide
- International Development Guide

Developer's Guide
- DOS and OS/2 Development Guide
- STREAMS Primer
- STREAMS Programmer's Guide
- STREAMS Network Programmer's Guide

CodeView and Macro Assembler User's Guide
- The CodeView Debugger
- Macro Assembler User's Guide

Device Driver Writer's Guide
- Writing, compiling, and linking drivers
- SCSI drivers
- STREAMS and line disciplines
- (K) Kernel routines

To order any of the above manuals, call 408/434–6688, ext. 3004 and give the manual title and part number.

Operating System Documents for Different Audiences

As shown on the previous page, Altos offers many manuals with Altos UNIX System V—the manuals you receive will depend on your configuration. To help you decide which manuals are best suited to your needs, we have listed below the manuals according to three broad groups of users.

These lists are only suggested starting points in your search for information. They are not meant to imply that certain users should *not* read certain manuals. Find the user group that best applies to you, and use its list of manuals as a starting point for your reading, from which you can move on to other manuals.

Note that every Run–time System includes five manuals: the *Installation Guide*, the *User's Guide*, the *User's Reference*, the *System Administrator's Guide*, and the *System Administrator's Reference*. The Run–time System reference pages that describe the C, M, F, ADM, and HW commands ("man pages") are provided online as well. If you have the Development System, all manuals listed under "For Programmers:" come with your operating system. (All Development System reference pages are also provided online.) To order additional manuals, call (408) 434–6688, extension 3004 and give the manual title and part number.

For General Users (especially Beginners):

Tutorial
User's Guide
User's Reference (C, M, F)
Using the AOM Menu System

For System Administrators (and Advanced Users):

Installation Guide
System Administrator's Guide
System Administrator's Reference (ADM, HW)
International Operating System Guide
Programmer's Reference (CP, S)

For Programmers:

Programmer's Guide
Programmer's Reference (CP, S)
C Language Guide
Library Guide
Developer's Guide
CodeView and Macro Assembler User's Guide
Device Driver Writer's Guide

Preface

Throughout the documentation, a given command, routine, or file is referred to by its name and a section (in parentheses). For example, the programming command *cc*, is listed as *cc*(CP), which indicates that *cc* is described in the Programming Commands (CP) section.

There is a total of twelve reference sections in Altos UNIX System V, in different volumes of the Operating System and the Development System documents. (These reference sections are often called *manual pages*, or just *man pages*, in short.) For example, the *cc*(CP) command mentioned above is located in the CP section found in the *Programmer's Reference*.

This document, the *System Administrator's Reference (ADM, HW)*, is a companion to the *System Administrator's Guide* and contains all commands that are reserved for exclusive use by system administrators. This manual contains the following two reference sections:

Section	Description	Volume
ADM	Administrative Commands - used for system administration.	System Administrator's Reference
HW	Hardware device manual pages - information about hardware devices and device nodes.	System Administrator's Reference

The following table lists the remaining reference sections, the type of commands they contain, and in which document each is located.

Section	Description	Volume
C	Commands - used with the Operating System.	User's Reference

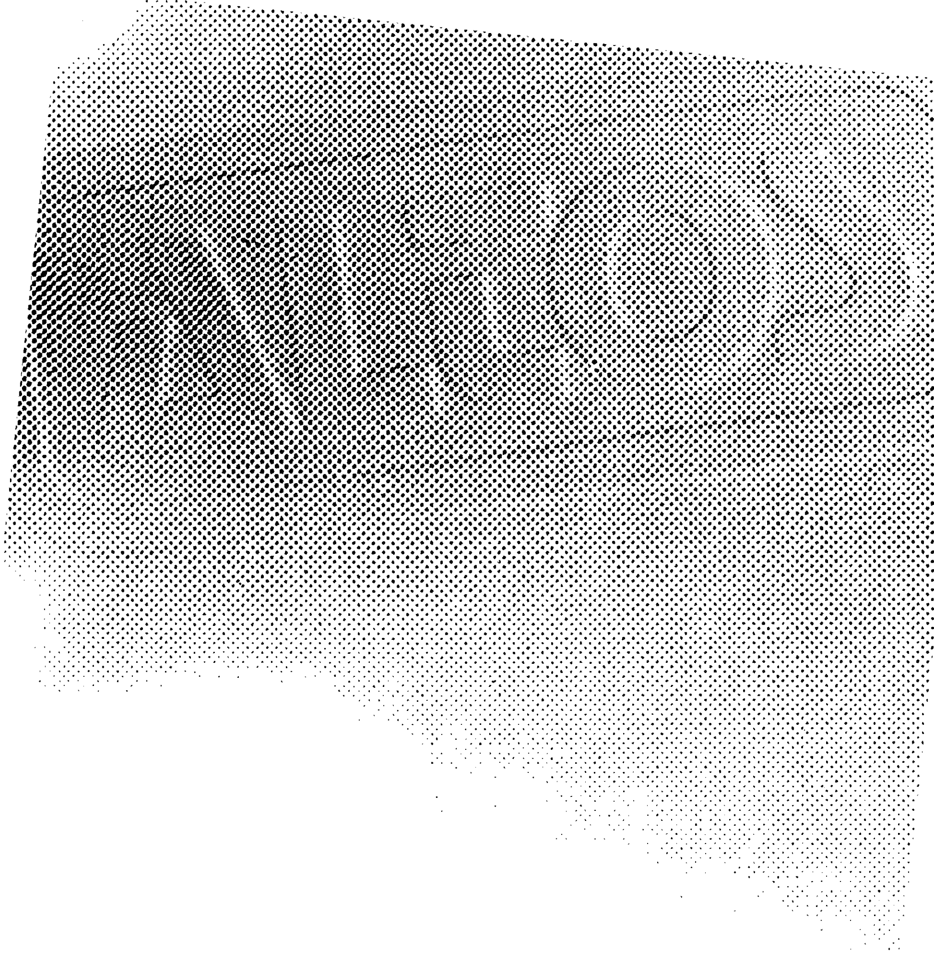
Section	Description	Volume
CP	Programming Commands - used with the Development System.	Programmer's Reference
DOS	DOS Cross-development subroutines and libraries	Developer's Guide
F	File Formats - description of various system files not defined in section M.	User's Reference
K	Kernel routines - used for writing device drivers.	Device Driver Writer's Guide
M	Miscellaneous - information used for access to devices, system maintenance, and communi-cations.	User's Reference
NSL	Network Services Library - used with the STREAMS System.	Developer's Guide
S	System Calls and Library Routines - available for C and assembly language programming.	Programmer's Reference
STR	STREAMS manual pages	Developer's Guide
XNX	XENIX cross-development manual pages	Library Guide

The alphabetized table of contents following this preface lists all Altos UNIX System V commands, system calls, library routines, and file formats. In addition, in the front of each individual reference section there is an alphabetized list of all the manual pages contained in that section.

The permuted index, found at the end of all Reference manuals, is useful in matching a desired task with the manual page that describes it. It too is an organized list of all Altos UNIX System V commands, system calls, library routines, and file formats, but organized according to function, not alphabetically. Note that some pages in the Operating System documents refer to "include" files that are actually part of the Development System.

Alphabetized List

Commands, Systems Calls, Library Routines and File Formats



Altos UNIX® System V/386
Release 3.2

(ADM) Administration

Contents

System Administration **(ADM)**

auditsh	menu driven audit administration utility
authck	check internal consistency of Authentication database
authsh	administrator interface for authorization subsystem
autoboot	automatically boots the system
backup	performs UNIX backup functions
backupsh	menu driven backup administration utility
badtrk	scans fixed disk for flaws and creates bad track table
brc, bcheckrc	system initialization procedures
captoinfo	convert a termcap description into a terminfo description
checkaddr	MMDF address verification program
checkque	MMDF queue status report generator
checkup	Report on MMDF problems
chg_audit	enables and disable auditing for the next session
chroot	changes root directory for command
cleanque	send warnings and return expired mail
cleantmp	remove temporary files in directories specified
clri	clears inode
configure	kernel configuration program
consoleprint	print /usr/adm/messages or any file to a serial printer attached to the printer port of a serial console
crash	examine system images
custom	installs specific portions of the UNIX System
dbmbuild	builds the MMDF hashed database of alias and routing information
dcopy	copy UNIX filesystems for optimal access time
deliver	MMDF mail delivery process
del.vd	delete a virtual disk
dial, uuchat	dials a modem
diskusg	generate disk accounting data by user ID
displaypkg	display installed packages
divvy	disk dividing utility
dlayout	display hard disk partition, division, and size information
dlvr_audit	produce audit records for subsystem events
dmesg	displays the system messages on the console
dparam	displays/changes hard disk characteristics
fdisk	maintain disk partitions
fdswap	swaps default boot floppy drive
ff	list file names and statistics for a filesystem
fixperm	correct or initialize file permissions and ownership
fsave	interactive, error-checking filesystem backup
fsck, dfsck	checks and repairs filesystems

fsdb	filesystem debugger
fsname	prints or changes the name of a file system
fsphoto	performs periodic semi-automated system backups
fsstat	report file system status
fstyp	determine file system identifier
fwtmp, wtmpfix	manipulate connect accounting records
goodpw	check a password for non-obviousness
graph	draws a graph
haltsys, reboot	closes out the file systems and shuts down the system
hdutil	hard disk utility for displaying and removing specific disk device names
id	print user and group IDs and names
idaddld	add or remove line disciplines from kernel configuration files
idbuild, idmkenv, idmkunix, idconfig, idvidi, idscsi	build new UNIX system kernel
idcheck	returns selected information
idinstall	add, delete, update, or get device driver configuration data
idleout	logs out idle users
idmemtune	adjusts tunable parameters to match system memory
idmkinit	read files containing specifications
idmknod	removes nodes and reads specifications of nodes
idspace	investigates free space
idtune	attempts to set value of a tunable parameter
infocmp	compare or print out terminfo descriptions
initcond	special security actions for init and getty
initscript	defines environment for programs executed by init(M)
install	install commands
installpkg	install package
integrity	examine system files against the authentication database
ipcrm	removes a message queue, semaphore set or shared memory ID
ipcs	reports the status of inter-process communication facilities
kbmode	set keyboard mode or test keyboard support
killall	kill all active processes
labelit	provide labels for filesystems
link, unlink	link and unlink files and directories
link_unix	builds a new UNIX system kernel

list	list processor channel for MMDF
lpadmin	configure the print service
lpfilter	administer filters used with the print service
lpforms	administer forms used with the print service
lpsched, lpshut, lpmove	start/stop the print service and move requests
lpsh	menu driven lp print service administration utility
lpusers	set printing queue priorities
majorsinuse	displays the list of major device numbers currently specified in the mdevice file
makekey	generates an encryption key
mkdev	calls scripts to add peripheral devices
mkfs	constructs a filesystem
mmdf	routes mail locally and over any supported network
mmdfalias	converts XENIX-style aliases file to MMDF format
mnlist	converts a XENIX-style Micnet routing file to MMDF format
mount	mounts and unmounts a file structure
mountall, umountall	mount, unmount multiple file systems
mvdir	moves a directory
ncheck	generates names from inode numbers
netutil	administers the micnet network
nictable	process NIC database into channel/domain tables
nlsadmin	network listener service administration
pcu	port configuration utility
profiler: prfld, prfstat, prfdc, prfsnap, prfpr	UNIX system profiler
proto	prototype job file for at, cron and batch
rc0	run commands performed to stop the operating system
rc2	run commands performed for multiuser environment
reduce	perform audit data analysis and reduction
relogin	rename login entry to show current layer
removepkg	remove installed package
restore	UNIX incremental filesystem backup restore
rmail	submit remote mail received via UUCP
routines	finds driver entry points in a driver object module
runacct	run daily accounting
sag	system activity graph
sar	system activity report package
schedule	database for automated system backups
scsinfo	display current SCSI device information

setclock	sets the system real-time (time of day) clock
setmnt	establishes /etc/mnttab table
settime	changes the access and modification dates of files
shutdown	terminates all processing
strace	prints STREAMS trace messages
strclean	STREAMS error logger cleanup program
strerr	STREAMS error logger daemon
strmcfg	STREAMS configuration utility for networking products
strmtune	STREAMS configuration interface for networking products
submit	MMDF mail enqueuer
sulogin	access single-user mode
swap	swap administrative interface
sync	updates the super-block
sysadmsh	menu driven system administration utility
sysdef	output values of tunable parameters
tcbck, smmck, authckrc	trusted computing base checker
timex	time a command; report process data and system activity
tplot	graphics filters
uadmin	administrative control
uconfig	UNIX configuration manager
umount	dismounts a file structure
upsconfig	UPS shutdown configuration utility
uucheck	checks the uucp directories and permissions file
uucico	file transport program for the UUCP system
uuclean	UUCP spool directory clean-up
uudemon: uudemon.admin, uudemon.clean, uudemon.hour, uudemon.poll, uudemon.poll2	UUCP administrative scripts
uugetty	set terminal type, modes, speed, and line discipline
uuinstall	administers UUCP control files
uulist	converts a UUCP routing file to MMDF format
uusched	the scheduler for the UUCP file transport program
uutry	tries to contact remote system with debugging on
uuxqt	executes remote command requests
vddaemon	virtual disk initialization
vdinfo	display virtual disk information
vdutil	virtual disk utility

vectorsinuse	displays the list of vectors currently specified in the sdevice file
volcopy	make literal copy of UNIX filesystem
wall	writes to all users
wtinit	object downloader for the 5620 DMD terminal
xbackup	performs XENIX incremental filesystem backup
xdumpdir	prints the names of files on a XENIX backup archive
xinstall	XENIX installation shell script
xprsetup	transparent printer setup utility
xrestore, xrestor	invokes XENIX incremental filesystem restorer
xtd	extract and print xt driver link structure
xts	extract and print xt driver statistics
xtt	extract and print xt driver packet traces

Intro

introduction to system administration commands

Description

This section contains the commands that are used to administrate and maintain the operating system. These commands are largely root-only, meaning that they can only be executed by the super-user (root).

accept, reject

allows/prevents print requests to a lineprinter or class of printers

Syntax

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

Description

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

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

-r[*reason*] Associates a *reason* with disabling (using *disable (C)) the printer. The reason* applies to all printers listed up to the next **-r** option. If the **-r** option is not present or the **-r** option is given without a *reason,* then a default *reason* is used. *Reason* is reported by *lpstat(C).* Please see *disable(C)* for an example of *reason* syntax.

Files

/usr/spool/lp/*

See Also

enable(C), lp(C), lpadmin(ADM), lpsched(ADM), lpstat(C), disable(C)

acct: acctdisk, acctdusg, accton, acctwtmp

overview of accounting and miscellaneous accounting commands

Syntax

/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. When the system is installed, accounting is initially in the "off" state. *acctsh* (ADM) 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*(F). The programs described in *acctcon* (ADM) convert this file into session and charging records, which are then summarized by *acctmerg* (ADM).

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 *acctprc* (ADM) summarize this data for charging purposes; *acctcms* (ADM) is used to summarize command usage. Current process data may be examined using *acctcom* (C).

Process accounting and connect time accounting [or any accounting records in the format described in *acct*(F)] can be merged and summarized into total accounting records by *acctmerg* [see **tacct** format in *acct*(F)]. *prtacct* [see *acctsh* (ADM)] 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 file names 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*(ADM) 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*(S) and *acct*(F)].

acctwtmp writes a *utmp*(F) 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*(F)]. *Reason* must be a string of 11 or fewer characters, numbers, $, or spaces. For example, the following are suggestions for use in reboot and shutdown procedures, respectively:

```
acctwtmp uname >> /etc/wtmp
acctwtmp "file save" >> /etc/wtmp
```

Files

/etc/passwd	used for login name to user ID conversions
/usr/lib/acct	holds all accounting commands listed in this manual
/usr/adm/pacct	current process accounting file
/etc/wtmp	login/logoff history file

See Also

acctcms(ADM), acctcom(C), acctcon(ADM), acctmerg(ADM), acctprc(ADM), acctsh(ADM), diskusg(ADM), fwtmp(ADM), runacct(ADM), acct(S), acct(F), utmp(F)

Standards Conformance

acctdisk is conformant with:

AT&T SVID Issue 2, Select Code 307-127.

Value Added

accton is an extension to AT&T System V provided in Altos UNIX System V.

acctcms

command summary from per-process accounting records

Syntax

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

Description

acctcms reads one or more *files*, normally in the form described in *acct*(F). 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*(C). 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 file names 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 previoustotal
acctcms -s today previoustotal >total
acctcms -a -s today
```

See Also

acct(ADM), acctcom(C), acctcon(ADM), acctmerg(ADM), acctprc(ADM), acctsh(ADM), fwtmp(ADM), runacct(ADM), acct(S), acct(F), utmp(F)

Notes

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.

Standards Conformance

acctcms is conformant with:

AT&T SVID Issue 2, Select Code 307-127.

acctcom

search and print process accounting file(s)

Syntax

acctcom [[options] [file]] . . .

Description

acctcom reads *file*, the standard input, or **/usr/adm/pacct**, in the form described by *acct*(F) and writes selected records to the standard output. Each record represents the execution of one process. The output shows the **COMMAND Name, USER, TTYName, START TIME, END TIME, REAL (SEC), CPU (SEC), MEAN SIZE(K)**, and optionally, **F** (the *fork/exec* flag: **1** for *fork* without *exec*), **STAT** (the system exit status), **HOG FACTOR, KCORE MIN, CPU FACTOR, CHARS TRNSFD**, and **BLOCKS READ** (total blocks read and written).

The command name is prepended with a **#** if it was executed with *super-user* privileges. If a process is not associated with a known terminal, a **?** is printed in the **TTYName** field.

If no *files* are specified, and if the standard input is associated with a terminal or **/dev/null** (as is the case when using **&** in the shell), **/usr/adm/pacct** is read; otherwise, the standard input is read.

If any *file* arguments are given, they are read in their respective order. Each file is normally read forward, i.e., in chronological order by process completion time. The file **/usr/adm/pacct** is usually the current file to be examined; a busy system may need several such files of which all but the current file are found in **/usr/adm/pacct***?*. The *options* are:

-a Show some average statistics about the processes selected. The statistics will be printed after the output records.

-b Read backwards, showing latest commands first. This *option* has no effect when the standard input is read.

-f Print the *fork/exec* flag and system exit status columns in the output.

-h Instead of mean memory size, show the fraction of total available CPU time consumed by the process during its execution. This "hog factor" is computed as:

(total CPU time)/(elapsed time).

-i Print columns containing the I/O counts in the output.

-k	Instead of memory size, show total kcore-minutes.
-m	Show mean core size (the default).
-r	Show CPU factor (user time/(system-time + user-time).
-t	Show separate system and user CPU times.
-v	Exclude column headings from the output.
-l *line*	Show only processes belonging to terminal **/dev/***line*.
-u *user*	Show only processes belonging to *user* that may be specified by: a user ID, a login name that is then converted to a user ID, a # which designates only those processes executed with *super-user* privileges, or ? which designates only those processes associated with unknown user IDs.
-g *group*	Show only processes belonging to *group*. The *group* may be designated by either the group ID or group name.
-s *time*	Select processes existing at or after *time*, given in the format *hr* [:*min* [:*sec*]].
-e *time*	Select processes existing at or before *time*.
-S *time*	Select processes starting at or after *time*.
-E *time*	Select processes ending at or before *time*. Using the same *time* for both **-S** and **-E** shows the processes that existed at *time*.
-n *pattern*	Show only commands matching *pattern* that may be a regular expression as in *ed*(C) except that + means one or more occurrences.
-q	Do not print any output records; just print the average statistics as with the **-a** option.
-o *ofile*	Copy selected process records in the input data format to *ofile*; suppress standard output printing.
-H *factor*	Show only processes that exceed *factor*, where factor is the "hog factor" as explained in option **-h** above.
-O *sec*	Show only processes with CPU system time exceeding *sec* seconds.
-C *sec*	Show only processes with total CPU time, system plus user, exceeding *sec* seconds.
-I *chars*	Show only processes transferring more characters than the cut-off number given by *chars*.

Files

/etc/passwd
/usr/adm/pacct
/etc/group

See Also

acct(ADM), acctcms(ADM), acctcon(ADM), acctmerg(ADM), acctprc(ADM), acctsh(ADM), fwtmp(ADM), ps(C), runacct(ADM), su(ADM), acct(S), acct(F), utmp(F).

Notes

acctcom reports only on processes that have terminated; use *ps*(C) for active processes. If *time* exceeds the present time, then *time* is interpreted as occurring on the previous day.

acctcon: acctcon1, acctcon2

connect-time accounting

Syntax

/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. Hang-up, termination of *login*(M) 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*(M) and *utmp*(F).

-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*(F)].

Examples

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

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

Files

/etc/wtmp

See Also

acct(ADM), acctcms(ADM), acctcom(C), acctmerg(ADM), acctprc(ADM), acctsh(ADM), fwtmp(ADM), init(M), runacct(ADM), acct(S), acct(F), utmp(F)

Notes

The line usage report is confused by date changes. Use *wtmpfix* [see *fwtmp*(ADM)] to correct this situation.

Standards Conformance

acctcon1 and *acctcon2* are conformant with:

AT&T SVID Issue 2, Select Code 307-127.

acctmerg

merge or add total accounting files

Syntax

/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*(F)] 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(ADM), acctcms(ADM), acctcom(C), acctcon(ADM), acctprc(ADM), acctsh(ADM), fwtmp(ADM), runacct(ADM), acct(S), acct(F), utmp(F)

Standards Conformance

acctmerg is conformant with:

AT&T SVID Issue 2, Select Code 307-127.

accton

turns on accounting

Syntax

accton [file]

Description

accton turns on and off process accounting. If no *file* is given then accounting is turned off. If *file* is given, the kernel appends process accounting records. (See *acct* (S) and *acct* (F)).

Files

/etc/passwd	Used for login name to user ID conversions
/usr/adm/pacct	Current process accounting file
/usr/adm/sulogin	Super-user login history file
/etc/wtmp	Login/logout history file

See Also

acctcom(ADM), acct(S), acct(F), su(C), utmp(F)

Value Added

accton is an extension to AT&T System V developed in Altos UNIX System V.

acctprc: acctprc1, acctprc2

process accounting

Syntax

/usr/lib/acct/acctprc1 [ctmp]

/usr/lib/acct/acctprc2

Description

acctprc1 reads input in the form described by *acct*(F), 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*(ADM), 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(ADM), acctcms(ADM), acctcom(C) acctcon(ADM), acctmerg(ADM), acctsh(ADM), cron(C), fwtmp(ADM), runacct(ADM), acct(S), acct(F), utmp(F)

Notes

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*(C), for example. More precise conversion can be done by faking login sessions on the console via the *acctwtmp* program in *acct*(ADM).

Standards Conformance

acctprc1 and *acctprc2* are conformant with:

AT&T SVID Issue 2, Select Code 307-127.

acctsh: chargefee, ckpacct, dodisk, lastlogin, monacct, nulladm, prctmp, prdaily, prtacct, runacct, shutacct, startup, turnacct

shell procedures for accounting

Syntax

/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*(C). 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 collection 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/default/filesys** . If the **-o** flag is used, it will do a slower version of disk accounting by login directory. *Files* specify the one or more filesystem names where disk accounting will be done. If *files* are used, disk accounting will be done on these file systems only. If the **-o** flag is used, *files* should be mount points of mounted filesystem. If omitted, they should be the special file names of mountable file systems.

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*(C) 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* (ADM).

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***mmdd* where *mmdd* 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 *mmdd* option and specifying the exact report date desired. The **-l** 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*(ADM).

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

startup is called by **/etc/init.d/acct** to turn the accounting on whenever the system is brought to a multiuser state.

turnacct is an interface to *accton* [see *acct*(ADM)] 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/pacct***incr* (where *incr* is a number starting with **1** 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/fee	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 (ADM)
/usr/adm/acct/sum	summary directory, should be saved

See Also

acct(ADM), acctcms(ADM), acctcom(ADM), acctcon(ADM), acctmerg(ADM), acctprc(ADM), cron(C), diskusg(ADM), fwtmp(ADM), runacct(ADM), acct(S), acct(F), utmp(F)

Standards Conformance

chargefee is conformant with:

ANSI X3.159-198X C Language Draft Standard, May 13, 1988.

ckpacct, *lastlogin*, *prctmp*, *runacct* and *shutacct* are conformant with:

AT&T SVID Issue 2, Select Code 307-127.

add.vd

add a virtual disk

Syntax

add.vd [-d] [*n*]

Description

The *add.vd* utility is not intended for direct use by system users. It is invoked by *vdutil*(ADM). *add.vd* installs a virtual disk (for building a striped or mirrored filesystem across several disk divisions or physical disks). The system must be in singleuser mode to use this command. You must have already run *mkdev hd* to add all the physical hard disks on the system before using *add.vd.* (See "Adding Hard Disks" in the *System Administrator's Guide.*)

If *n* is not specified on the command line, the script prompts you for the virtual disk number. Once you reply with a correct number, the system installs the designated virtual disk.

Several questions are asked interactively to identify the physical disk divisions to be used and the mount point.

To efficiently use the available disk space, all disk divisions that comprise the virtual disk should be the same size.

Options

-d Non-destructive mode. The device special files are created, but no new filesystem is built.

n Add virtual disk number *n*.

See Also

del.vd(ADM, vdinfo(ADM), vddaemon(ADM) vdutil(ADM)

Note

Also refer to "Virtual Disks" in the *System Administrator's Guide.*

Value Added

add.vd is an extension of AT&T System V provided in Altos UNIX System V.

addxusers

create new user accounts given a XENIX-style password file

Syntax

/tcb/bin/addxusers [**-e**] [**-s**] [**-t** type] [file]

Description

addxusers reads the specified *file*, which should be in XENIX *passwd*(F) format, and creates the indicated accounts by making equivalent entries in the system's **/etc/passwd** file and Protected Password database. The **auth** subsystem and **chown** kernel authorizations are required to run *addxusers*. If no *file* is given, *addxusers* does not attempt to add any new users and only performs certain consistency checks on the existing user accounts. A *file* of **-** means that the standard input should be read.

Login names must begin with a lower case letter, must not already exist, must not contain a slash (“/”), and must not be longer than 8 characters.

Numeric user IDs must not be already assigned, and must be in the range 0 to 60000 (inclusive).

Numeric group IDs must be in the range 0 to 60000 (inclusive). Groups which are missing from the file **/etc/group** are warned about, as is membership in a group associated with a protected subsystem.

Encrypted passwords are preserved; that is, users will be able to use their old XENIX passwords to log onto the new system.

Any password aging information which is present is translated into the equivalent expiration parameters.

The comment field, initial working directory (home directory), and shell program are preserved. Missing or inaccessible directories and shells are warned about, as are non-absolute pathnames. Users should not share home directories.

The **-t** option sets the *type* of each created user; if omitted, each user is classified as an "individual" person. The legal *type* values are:

User *type* values			Comments
Number	Equivalent names		
0	root	superuser	All-powerful user (numeric ID 0).
1	operator		Various classifications of anonymous system administration accounts.
2	sso	security officer	
3	admin	administrator	
4	pseudo	pseudo-user	General-purpose anonymous user.
5	general	individual	A human's personal account.
6	retired		An account which is no longer used.

A "retired" user cannot log in and cannot be un-retired. No user may *su*(C) to an "individual" account.

Normally, only minimal checks for corruption are done on the existing **/etc/passwd** file before the new users are added: Checks are only done for duplicated login names or numeric user IDs, and bad format. (These are all fatal errors, and prevent any new users from being added.) The **-e** option causes the same checks which are applied to new users to be applied to the existing users (except for membership in a protected subsystem group). The **-s** option checks the existing users for being a member of a protected subsystem group. As with new user accounts, not all of the problems which may be discovered are fatal (many are only warnings).

Duplicated group names or numeric group IDs in the **/etc/group** file are warned about. However, if a protected subsystem group is so corrupted, this is a fatal error (no users are added).

Example

The following steps should be performed when migrating a community of users from a XENIX system:

1. Back up the home directories of the users on the XENIX system using *cpio*(C) or *tar*(C).

2. Make a copy of **/etc/passwd** and **/etc/group** from the XENIX system. (Do not back up these files using absolute pathnames. For example, if your accounts are in **/usr**, run your backup command from that directory, not from **/**.)

3. After making certain you are in single user mode, extract the backup of the user's home directories on the new system. For example, if your user accounts reside in **/usr**, the files should be extracted in **/usr** on the new system. (Note that if you are using **/u** for your accounts, you must mount it before extracting your backups.)

4. Extract the copy of the **passwd** and **group** files in a temporary directory; for example, **/tmp/passwd** and **/tmp/group**. Be careful not to overwrite the **/etc/passwd** and **/etc/group** files on the new system.

5. Edit **/tmp/passwd** to remove "system" accounts (such as **root** and **bin**) and any accounts that already exist on the new system.

6. Separate the remaining accounts in **/tmp/passwd** (which are to be added to the new system) into different files by user type. For example, place all "pseudo-users" in a file called **/tmp/pseudo** and all "individual" humans in **/tmp/individual.**

7. In your sorted **/tmp** account files, you should then change login names, numeric user IDs, numeric group IDs, initial working directories, and shell programs as necessary to prevent conflicts with any accounts already on the new system. (If any numeric user or group IDs are changed, it may be desirable to *chown*(C) or *chgrp*(C) the appropriate home directories on the new system and their contents.)

8. Merge **/tmp/group** (the saved copy of the XENIX system's **/etc/group**) with the new system's **/etc/group**; see *group*(F). Again, make certain you are still in single-user mode; if **/etc/group** is modified while in multi-user mode, no-one will be allowed to log in.

9. Run *addxusers* :

   ```
   addxusers -t pseudo-user /tmp/pseudo 2>&1 | tee -a /tmp/errors
   addxusers -t individual /tmp/individual 2>&1 | tee -a /tmp/errors
   ...
   ```

   It is advisable to save the standard output and error output of *addxusers* (as shown above) for later analysis and correction.

Finally, use the **Accounts→User→Examine** menu of *sysadmsh*(ADM) to customize the newly-created accounts as needed.

The authorizations may need customization, and accounts which are neither individuals nor retired should have an "account which may su" assigned.

See Also

authcap(F), chgrp(C), chown(C), cpio(C), group(F), passwd(F), su(C), sysadmsh(ADM), tar(C), tee(C)

Notes

When logging in, XENIX truncates passwords to eight (8) characters;

Altos UNIX System V does not. Therefore, the user must not type more than eight characters when the password from the XENIX system is in effect.

Passwordless accounts and other liberties XENIX allows are more restricted in Altos UNIX System V. To continue to use such poor security practices requires customizing the system defaults or the insecure accounts.

Some standard accounts shipped with the system provoke warnings when the **-e** or **-s** options are specified.

Some vendor's systems support specifying a *nice*(S) value in the comment field, or doing a *chroot*(S) to the home directory (called a sublogin). Both constructions are understood by *addxusers*, albeit sublogins are not supported in Altos UNIX System V and cause a warning.

adfmt

formats SCSI hard disks

Syntax

/etc/**adfmt** device_name

Description

The **adfmt** command issues a **format** command to the SCSI disk *device_name*. *device_name* should be the character-special device representing the whole SCSI disk, for example, */dev/rhd00*.

Notes

SCSI disks with embedded controllers are formatted as part of the manufacturing test procedure. Using *adfmt* on these disks is unnecessary.

Files

/dev/rhd?0

See Also

scsi(HW), hd(HW)

Value Added

adfmt is an extension of AT&T System V provided in Altos UNIX System V.

asktime

prompts for the correct time of day

Syntax

/etc/asktime

Description

This command prompts for the time of day. You must enter a legal time according to the proper format as defined below:

[[*yy*]*mmdd*]*hhmm*

Here the first *mm* is the month number; *dd* is the day number in the month; *hh* is the hour number (24-hour system); the second *mm* is the minute number; *yy* is the last 2 digits of the year number and is optional. The current year is the default if no year is mentioned.

Examples

This example sets the new time, date, and year to "11:29 April 20, 1985".

```
Current system time is Wed Nov  3 14:36:23 PST 1985
Enter time ([yymmdd]hhmm): 8504201129
```

Diagnostics

If you enter an illegal time, *asktime* prompts with:

```
Try again:
```

Notes

asktime is normally performed automatically by the **/etc/rc2** system startup scripts immediately after the system is booted; however, it may be executed at any time. The command is privileged, and can only be executed by the super-user.

Systems which autoboot will invoke *asktime* automatically on reboot. On these systems, if you don't enter a new time or press return within 1 minute of invoking *asktime* , the system will use the time value it has. If RETURN alone is entered, the time is unchanged.

Value Added

asktime is an extension of AT&T System V provided in Altos UNIX System V.

atcronsh

at and cron administration utility

Syntax

/usr/lib/sysadm/atcronsh

Description

atcronsh is the screen interface invoked by the *sysadmsh*(ADM) Jobs→Authorize selection. It is used to specify users allowed to use the *cron*(C), *at*(C) and *batch*(C) commands. It also allows the *at*(C) and *batch*(C) prototype files to be edited.

The program allows a system default for both *cron*(C) and *at*(C) and *batch*(C) to be given. The defaults can be:

- none - No user authorized
- allow - All users allowed to use the commands unless a user is specifically denied
- deny - All users denied to use the commands unless a user is specifically authorised

The default setting decides whether an allow or deny file is to be used (deny file means **/usr/lib/cron/cron.deny** or **at.deny**, allow file means **/usr/lib/cron/cron.deny** or **at.deny**).

For each user (unless the none system default has been chosen), a specific authorization for both *cron*(C) and *at*(C) and *batch*(C) may be given. The allow and deny files are interpreted as follows:

- if an allow file exists, and the user name appears in it, the user is allowed access.
- if an allow file exists, access is denied
- if a deny file exists and the user name appears in it, access id denied
- if a deny file exists, access is allowed
- access is denied

Files

/usr/lib/cron/cron.allow
/usr/lib/cron/cron.deny
/usr/lib/cron/at.allow
/usr/lib/cron/at.deny

See Also

auditsh(ADM), authsh(ADM), at(C), backupsh(ADM), batch(C), cron(C), lpsh(ADM), sysadmsh(ADM)

Notes

Invoking *atcronsh*(ADM) is not recommended; use the *sysadmsh*(ADM) Jobs→Authorize selection.

Value Added

atcronsh is an extension of AT&T System V provided in Altos UNIX System V.

auditcmd

command interface for audit subsystem activation, termination, statistic retrieval, and subsystem notification

Syntax

auditcmd [**-e**] [**-d**] [**-s**] [**-c**] [**-m**] [**-q**]

Description

The *auditcmd* utility is used to control the audit subsystem. This command may only be executed by processes with the **configaudit** kernel authorization since the audit device is used.

auditcmd allows the specification of one of the following options:

-e Enable the audit subsystem for audit record generation. The enabling of the audit subsystem initializes subsystem parameters from the **/tcb/files/audit/audit_parms** file. This file is established using the *auditif*(ADM) command.

-s Inform the audit subsystem that a system shutdown is in progress. The subsystem will continue audit record generation to a temporary directory on the root file system. The audit daemon is also modified so that it will survive the shutdown. The subsystem will continue to generate audit records until disabled.

-d Disable the audit subsystem. All audit record generation ceases and a termination record is written to the audit trail. This record results in the termination of the audit daemon. The subsystem properly synchronizes to insure that the audit daemon has read all records from the audit trail before the system is allowed to terminate.

-m Inform the audit subsystem that multi-user run state has been achieved and that alternate audit directories specified by the administrator using *auditif* are now mounted and available.

-c Retrieve audit subsystem statistics from the audit device.

-q Perform the specified option silently. Do not report errors attributable to the audit subsystem not being enabled at the moment.

See Also

audit(HW), "Maintaining System Security," chapter of the *System Administrator's Guide*

Diagnostics

auditcmd returns 0 on success, 1 on command line argument error, and -1 on failure actions. Reasons for failure include parameter file inconsistencies, lack of permission, and security database inconsistency.

Value Added

auditcmd is an extension of AT&T System V provided in Altos UNIX System V.

auditd

read audit collection files generated by the audit subsystem and compact the records

Syntax

auditd [**-y**] [**-n**]

Description

auditd is the audit daemon process which is spawned whenever the audit subsystem is enabled. The audit subsystem continually generates audit records writing them to intermediate files called audit collection files. At any time, there may be many collection files since the subsystem continually switches files to ensure that no single file grows excessively large.

The daemon is responsible for reading the audit collection file records from the subsystem, compacting them to provide space savings, and writing the compacted records to files which will later be used for reduction. To read the records from the subsystem, the daemon uses the **/dev/auditr** device. The daemon exclusively reads this file which is managed by the subsystem. Each read request returns a block of data from a collection file. The audit subsystem insures that the data is returned in the proper order and also handles file management associated with the multiple collection files. This provides the daemon with a single read focal point.

As a block of data is returned to the daemon, it is optionally compacted and the record along with its size prepended is written to the current audit output file. Like the audit subsystem, the daemon is capable of writing many different output files in a number of administrator specified directories to avoid overflowing any one file system. As each output file is written, the daemon records the name in a log file which is used by the reduction program. This log file provides an output file trail alleviating the need for the administrator to keep up with file generation or to recreate the sequence of output file writing. The compaction of output files and the selection of audit directories is controlled by the administrator interface utility *auditsh*(ADM).

Each time the audit subsystem is enabled, a new audit session is created. The session is identified by a session ID which is used to stamp the output files generated by the audit daemon and the log file that identifies them. *auditif* is used to examine daemon log files in the **/tcb/files/audit** directory to identify the session and the date/time of the start and end of the session. In this manner, the administrator need not know the session ID but only the dates for which data reduction is desired.

When the daemon is started, a recovery mechanism is invoked to determine if the previous audit session was terminated normally. If abnormal termination occurred, there may be audit records written by the subsystem to collection files that were not read by the daemon and compacted to an audit output file. The daemon recovery mechanism provides the capability to recover these records and update the output files from the previous session as necessary. The recovery mechanism will interactively query whether recovery is desired if abnormal termination occurred. The **-y** and **-n** options may be to used avoid the interactive question.

The daemon also provides a mechanism whereby applications that are not privileged to open and write audit records to the audit device are able to send the daemon audit records. These are, in turn, written to to the audit subsystem. To provide this service, the daemon creates a message queue which only certain applications with specific permission are able to send messages to. When one of the applications wishes to generate an audit record using this mechanism, the record is first constructed and then written to the message queue. The specific message queue is identified in the file **/tcb/files/audit/audit_dmninfo.** This file contains the *audit_dmninfo* structure which is defined in the include file **sys/audit.h.** The first field is the process ID of the daemon and the second is the message queue identifier. After the message has been written to the queue by the application, the application will generate a **SIGUSR1** to the daemon indicating a message is waiting. The daemon responds by reading the message queue and writing the record to the audit subsystem device.

Files

/dev/auditr
/dev/auditw
/tcb/files/audit/audit_dmninfo
/tcb/files/audit/CAFLOG.xxxxxx

See Also

audit(HW), "Maintaining System Security," chapter of the *System Administrator's Guide*

Diagnostics

Upon successful completion at the termination of auditing by the subsystem, the program exits with a status of 0. Otherwise, a diagnostic message is printed and the program exits with a status of -1.

Value Added

auditd is an extension of AT&T System V provided in Altos UNIX System V.

auditsh

menu driven audit administration utility

Syntax

/usr/lib/sysadm/auditsh

Description

auditsh is the screen interface invoked by the *sysadmsh*(ADM) System→Audit selection. This selection controls the audit subsystem, allowing establishment of audit subsystem initialization parameters, specification of criteria for selecting output records during reduction, report generation, dynamic changing of subsystem parameters, and backup and restore of compacted audit output files.

If the environment variable **PAGER** is set, the specified program is used to display reports sent to the terminal.

See Also

atcronsh(ADM), auditcmd(ADM), auditd(ADM), authsh(ADM), backupsh(ADM), lpsh(ADM), reduce(ADM), sysadmsh(ADM)

Notes

Invoking *auditsh*(ADM) is not recommended; use the *sysadmsh*(ADM) System→Configure→Audit selection.

Value Added

auditsh is an extension of AT&T System V provided in Altos UNIX System V.

authck

check internal consistency of Authentication database

Syntax

authck [-p] [-t] [-s] [-f] [-c] [-a] [-v]

Description

authck checks both the overall structure and internal field consistency of all components of the Authentication database. It reports all problems it finds. The options and tests are as follows:

-p Check the Protected Password database. A number of tests are performed. The Protected Password and **/etc/passwd** are checked for completeness such that neither contains entries not in the other. Once this is done, the fields common to the Protected Password database and **/etc/passwd** are checked to make sure they agree. Then, fields in the Protected Password database are checked for reasonable values. For instance, all time stamps of past events are checked to make sure they have times less than that returned by *time*(S).

-t The fields in the Terminal Control database are checked for reasonable values. All time stamps of past events are checked to make sure they have times less than returned by *time*.

-s The Protected Subsystem database files are checked to ensure they correctly reflect the subsystem authorization entries in the Protected Password database. Each name listed in each subsystem file is verified against the Protected Password entry with the same name, so that no authorization is inconsistent between the files. Also, each Protected Password entry is scanned to ensure that all the privileges listed do in fact get reflected in the Protected Subsystem database. If any inconsistencies are found, the administrator is given the option of fixing the Subsystem database automatically.

-a This option is shorthand for turning on all the **-p**, **-t**, and **-s**, options.

-v This options provides running diagnostics as the program proceeds. It also produces warnings on events that should not occur but otherwise do not harm the Authentication database and the routines operating on it.

Files

/etc/passwd - System password file
/tcb/files/auth/*/* - Protected Password database
/etc/auth/system/ttys - Terminal Control database
/etc/auth/system/files - File Control database
/etc/auth/subsystems/* - Protected Subsystem database
/etc/auth/system/default - System Defaults database

See Also

integrity(ADM), getprpwent(S), getprtcent(S), getprfient(S), getprdfent(S), authcap(F), subsystem(S), "Maintaining System Security," chapter of the *System Administrator's Guide*

Value Added

authck is an extension of AT&T System V provided in Altos UNIX System V.

authsh

administrator interface for authorization subsystem

Syntax

/usr/lib/sysadm/authsh

Description

authsh is the screen interface invoked by the *sysadmsh*(ADM) Accounts selection to administer the authorization subsystem. It is a full screen menu driven interface that provides the functions necessary to control the generation and maintenance of user and system passwords, the terminal database configuration, terminal and account locking, and the generation of administrator reports on system activity.

The functions supported by the main level menu are:

User — This category of screen interfaces is provided for the setup and maintenance of user accounts and user account passwords. The screens are used to add, update, display, and delete user accounts from the system. Also, modifications to user account passwords or modifications to the various criteria controlling the generation of account passwords is accomplished using this menu option.

System — These options are provided for the maintenance of system-wide parameters like default privileges, password expiration, password lifetime, single user password requirement, restrictive password generation, and the delay time between login attempts. These parameters apply on a global system basis rather than a user account basis.

Terminals — The terminal database interface screens are used for the maintenance of the database entries to support the addition, deletion, and update of terminal information. Additionally, this category includes the necessary screens for setting and clearing locks on specific terminals.

Reports — This category provides the administrator with a method of generating various reports on system activity. Report types include password database, terminal database, and login activity reports.

The interface program is located in the trusted computing base directory **/tcb/bin.**

See Also

passwd(C), "Maintaining System Security," chapter of the *System Administrator's Guide*

Files

/etc/group,
/etc/passwd,
/tcb/files/auth/[a-z]/*
/tcb/files/biodb/[a-z]/*,
/tcb/files/biodb/schema,

Notes

Invoking *authsh*(ADM) is not recommended; use the *sysadmsh*(ADM) Accounts selection.

Value Added

authsh is an extension of AT&T System V provided in Altos UNIX System V.

autoboot

automatically boots the system

Description

The system can be set up to go through the boot stages automatically (as defined in **/etc/default/boot**) when the computer is turned on (booted), provided no key is pressed at the *boot*(HW) prompt.

If *boot* times out and AUTOBOOT=YES, then the word "auto" is passed in the boot string and *init*(M) is passed a **-a** flag.

In addition, the TIMEOUT entry can be set to specify the number of seconds to wait before timing out.

The *autoboot* procedure checks the file **/etc/default/boot** for the following instructions on autobooting:

AUTOBOOT=YES or NO	Whether or not *boot*(HW) times out and loads the kernel. *boot* looks for this variable in the **/etc/default/boot** file on its default device.
MULTIUSER=YES or NO	Whether or not *init*(M) invokes *sulogin* or proceeds to multiuser mode.
PANICBOOT=YES or NO	Whether or not the system reboots after a panic(). This variable is read from **/etc/default/boot** by *init*.
RONLYROOT=YES or NO	Whether or not the root filesystem is mounted *readonly*. This must be used only during installation, and not for a normal boot. It will effectively prevent writing to the filesystem.
DEFBOOTSTR=*bootstring*	Set default bootstring to *bootstring*. This is the string used by boot when the user presses <RETURN> only to the "Boot:" prompt, or when boot times out.

SYSTTY=*x*	If *x* is **1**, the system console device is set to the serial adapter at COM. If *x* is **0**, the system console is set to the main display adapter.
SLEEPTIME=*n*	Sets the time (in seconds) between calls to *sync*.
TIMEOUT=*n*	where *n* is the number of seconds to timeout at the "Boot:" prompt before booting the kernel (if AUTOBOOT=YES). If TIMEOUT is unspecified, defaults to one minute.

If either the **/etc/default/boot** file or the variable needed cannot be found, the variable is assumed to be NO. However, if the filesystem cannot be found, PANICBOOT is set to YES.

If the UNIX mail system, *mail*(C), is installed on the system, the output of the boot sequence is mailed to *root*. Otherwise, the system administrator should check the file **/etc/bootlog** for the boot sequence output. The output of *fsck*(ADM) is temporarily saved in the file **/dev/recover** before it is moved to **/etc/bootlog** and finally may be sent to the system administrator via *mail*.

Other boot options which take affect during *autoboot* are documented on the *boot*(HW) manual page.

Files

/etc/bootlog	*boot* output log for autobooting systems
/etc/default/boot	boot information file
/etc/rc2	instructions for entering multiuser mode, includes mounting and checking additional filesystems
/etc/sulogin	executed at startup, prompts the user to press Ctrl-d for multiuser mode or to enter the root password for maintenance mode
/dev/recover	allows saving of *fsck* output
/dev/scratch	temporary *fsck* file for large filesystems

See Also

boot(HW), fsck(ADM), init(M)

Notes

The utilities invoked during the boot procedure are passed the **-a** flag and time out only when the system *autoboots*. For example, *asktime*(ADM) times out after 30 seconds when the system *autoboots*, but waits for a response from the user any other time it is invoked.

The previous *boot* modes of AUTO=CLEAN, DIRTY, NEVER have been retained for backwards compatibility, but are ignored if any of the newer modes are present.

Value Added

autoboot is an extension to AT&T System V developed in Altos UNIX System V.

backup

performs UNIX backup functions

Syntax

backup [**-t**] [**-p** | **-c** | **-f** files | **-u** "user1 [user2]"] **-d** device

backup -h

Description

The UNIX backup utility is a front-end for the *cpio*(C) utility. Use *restore*(ADM) to restore backups made with this utility. It is not recommended for routine system backups; use the *sysadmsh*(ADM) interface for system backups.

-h produces a history of backups. Tells the user when the last complete and incremental/partial backups were done.

-c complete backup. All files changed since the system was installed are backed up.

-p incremental/partial backup. This option backs up only the files that have been modified since the date of the last backup. A complete backup must be done before a partial backup.

-f backup files specified by the <*files*> argument. File names may contain characters to be expanded (i.e., *, .) by the shell. The argument must be in quotes.

-u backup a user's home directory. All files in the user's home directory will be backed up. At least one user must be specified but it can be more. The argument must be in quotes if more than one user is specified. If the user name is "all", then all the user's home directories will be backed up.

-d used to specify the device to be used. It defaults to **/dev/rdsk/f0q15d** (the 1.2M floppy).

-t used when the device is a tape. This option must be used with the **-d** option when the tape device is specified.

A complete backup must be done before a partial backup can be done. Raw devices rather than block devices should always be used. The program can handle multi-volume backups. The program will prompt the user when it is ready for the next medium. The program will give you an estimated number of floppies/tapes that will be needed to do the backup. Floppies MUST be formatted before the backup is done. Tapes do not need to be formatted, except mini-cartridge tapes. If backup is done to tape, the tape must be rewound.

xbackup is the equivalent utility for XENIX filesystems.

backupsh

menu driven backup administration utility

Syntax

/usr/lib/sysadm/backupsh

Description

backupsh is the screen interface invoked by the *sysadmsh*(ADM) Backups selection to administer the backup subsystem. *backupsh* allows scheduled and non scheduled backups to be taken. Complete filesystems or single files or directories may also be restored. It also allows the **/usr/lib/sysadmin/schedule** file to be edited.

Backupsh can be used with both UNIX and XENIX filesystems. If a UNIX filesystem is being used then *backupsh* calls *cpio*(C), if a XENIX filesystem is being used then *backupsh* calls *xbackup*(ADM) or *xrestore*(ADM).

Refer *atcronsh*(ADM) for details of environment variables that *backupsh* uses, the usage is the same except that backpsh uses the specific variable **BACKUP** instead of **ATCRON**.

Files

/usr/lib/sysadmin/schedule

See Also

atcronsh(ADM) auditsh(ADM), authsh(ADM), at(C), batch(C), cpio(C), cron(C), backup(ADM), lpsh(ADM), restore(ADM), sysadmsh(ADM)

Notes

Invoking *backupsh*(ADM) is not recommended; use the *sysadmsh*(ADM) Backups selection.

Value Added

backupsh is an extension of AT&T System V provided in Altos UNIX System V.

badtrk

scans fixed disk for flaws and creates bad track table

Syntax

badtrk [**-e** [**-m** max]] [**-S** major# minor# blk# [#_of_blocks]]
[**-s qtdn**] [**-f** device]

Description

Used chiefly during system installation, *badtrk* scans the media surface for flaws, creates a new bad track table, prints the current table, and adds and deletes entries in the table. Bad tracks listed in the table are ''aliased'' to good tracks, such that when a process tries to read or write a track listed in the bad track table, one of a replacement tracks is used instead. These replacement tracks are allocated when *badtrk* is run during installation. Changing the number of replacement tracks allocated may require re-installation of the operating system, so the number of replacement tracks allocated should be fairly large.

To use *badtrk*, you must be in single user mode. (See *shutdown*(ADM)).

Note that SCSI disks maintain their own low-level record of bad blocks, without employing the operating system's bad block table or any of its aliasing schemes. Use the **-S** option to change the low-level SCSI disk record directly.

WARNING

badtrk operates correctly with SCSI disks when used with the **-S** option only. Since only SCSI disks are currently supported, running **badtrk** with any other option (or interactively with no options) may damage or corrupt disk data.

Options

-f *device*
Opens the partition *device* and reads the bad track table associated with that partition. *device* must be the active UNIX partition of a fixed disk: **/dev/rhdaa** for the first drive, **/dev/rhdba** for the second, and so on. The default is **/dev/rhdaa.**

-e Used by the installation procedure, the **-e** flag causes *badtrk* to change the size of the bad track table.

WARNING: The **-e** flag should not be invoked by the user. Use of the **-e** option may restructure the hard disk, rendering much of the information stored on it unusable.

-S *major# minor# block#* [*#_of_blocks*]
Re-assigns a bad block on a SCSI disk. The SCSI disk is identified by its major number *major#* and minor number *minor#*. The block number is identified by the *block#* argument. If more than this one block is to be identified, use the optional argument *#_of_blocks* to specify the total number of blocks starting at *block#*.

-m *max*
Used only in non-interactive mode in conjunction with **-e**, **-m** sets the maximum number of bad tracks to *max*.

-s *arguments*
Invokes *badtrk* non-interactively, causing it to scan the disk for bad tracks and enter any errors found in the bad track table. The *arguments* specify either quick or thorough, and either destructive or non-destructive scan:

[q]uick
[t]horough
[d]estructive
[n]on-destructive

The user should specify either **q** or **t**, and either **d** or **n**.

Usage

When *badtrk* is executed interactively, the program first displays the main menu:

```
1.   Print Current Bad Track Table
2.   Scan Disk (You can choose Read-Only or Destructive later)
3.   Add Entries to Current Bad Track Table by Cylinder/Head Number
4.   Add Entries to Current Bad Track Table by Sector Number
5.   Delete Entries Individually From Current Bad Track Table
6.   Delete All Entries From Bad Track Table

     Enter your choice or 'q' to quit:
```

You are prompted for option numbers, and, depending upon the option, more information may be queried for later.

A bad track table (option "1") might look like this:

Defective Tracks

	Cylinder	Head	Sector Number(s)
1.	190	3	12971-12987

```
Press <RETURN> to continue.
```

Option "2" scans the disk for flaws. If changes have been made to your bad track table since you last updated the table on disk (or since you entered *badtrk*), you will be asked if you want to update the disk with the new table before scanning. You should answer "y" to save your changes, 'n' if you don't want to save changes made up to this point. Next you are prompted to specify the kind of scan you wish to perform: either quick or thorough, and either destructive or non-destructive. Choosing a destructive scan will cause all data in the scanned region to be lost. After you respond to these prompts, *badtrk* begins its scan. You can interrupt a scan by typing "q" at any time. You are then prompted to continue the scan or return to the main menu.

As the program finds flawed tracks, it displays the location of each bad track. An example error message might be:

```
wd: ERROR : on fixed disk ctlr=0 dev=0/47 block=31434 cmd=00000020
    status=00005180, sector = 62899, cylinder/head = 483/4
```

(You may see this kind of message if there is a read or write error during the scanning procedure.)

When the scan is complete, the main menu reappears. The program automatically enters any detected flaws in the bad track table.

If your disk is furnished with a flaw map, you should enter these flaws into the bad track table. Select either option "3" or "4", depending upon the format of the flaw map furnished with your disk. Enter the defective tracks, one per line.

When you are satisfied that *badtrk* contains a table of the desired flaws, quit the *badtrk* program by entering "q" at the main menu.

If *badtrk* was invoked with the -e flag (which should only occur when called by *hdinit*, during the installation procedure), and the disk contains a valid division table, the following message is displayed prior to the *badtrk* menu:

```
This device contains a valid division table.  Additional
(non-root) filesystems can be preserved across this reinstallation.
If you wish to be able to preserve these file systems later, you must
not change the current limit of the bad track table, which is
n bad tracks.  Do you wish to leave it unchanged? <y/n>:
```

If you respond "y", you will not be prompted later to enter a new limit for the size of your bad track table. You can add or delete entries, but you will not be allowed to increase the maximum number of bad tracks allocated. If you respond "n" and the size of your bad track table is changed, your disk division table will be destroyed.

If you do not have a valid disk table or you selected "n" when prompted, you are prompted for the number of replacement tracks to allocate. There will be a recommended number of replacement tracks to allocate based on the number of known bad tracks plus an allowance for tracks that may go bad in the future. You should choose to allocate at least the recommended number of replacement tracks. Make your choice carefully, because if you want to change this amount later, you will have to reinstall.

Before exiting, *badtrk* will ask whether you wish to update the device with the new bad track table. If you wish to save you changes, answer "y". If you wish to leave the bad track table as it was before running *badtrk*, answer "n".

Notes

This utility can only be used in single-user mode.

If a bad spot develops in the boot blocks or system tables at the very beginning of the fdisk partition, reinstallation is required.

Files

/etc/badtrk

Value Added

badtrk is an extension of AT&T System V provided in Altos UNIX System V.

brc, bcheckrc

system initialization procedures

Syntax

/etc/bcheckrc [**-a**]

/etc/brc

Description

These shell procedures are executed via entries in **/etc/inittab** by *init*(M) 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. When invoked with the **-a** (autoboot) flag, *bcheckrc* will run without operator intervention. *init* calls *bcheckrc* with the **-a** flag when the system autoboots.

Then, the *brc* procedure clears the mounted file system table, **/etc/mnttab**, 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

boot(HW), fsck(ADM), init(M), rc2(ADM), shutdown(ADM)

captoinfo

convert a termcap description into a terminfo description

Syntax

captoinfo [**-v** ...] [**-V**] [**-1**] [**-w** width] file ...

Description

The *captoinfo* command looks in *file* for *termcap* descriptions. For each one found, an equivalent *terminfo*(F) 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 **TERMCAP** is used for the file name or entry. If **TERMCAP** is a full path name to a file, only the terminal whose name is specified in the environment variable **TERM** is extracted from that file. If the environment variable **TERMCAP** 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, up to a maximum width of 60 characters.

-w change the output to *width* characters.

Files

/usr/lib/terminfo/?/* compiled terminal description data base

Notes

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 cud1* and *ind*, respectively.) Padding information is assumed to belong at the end of the string.

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

The short two-letter name at the beginning of the list of names in a *termcap* entry, present for backwards compatibility, 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*(C) 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(ADM), tic(C), curses (S), terminfo(F)

checkaddr

MMDF address verification program

Syntax

/usr/mmdf/bin/checkaddr [-w] [*addresses...*]

Description

The *checkaddr* program is used to check the validity of an address within the local mail system (MMDF). *checkaddr* can be given addresses either on the command line, one address per argument, or a list of addresses can be given to *checkaddr* on the standard input, one address per line. The latter mode is use for checking the addresses in a mailing list by saying "*checkaddr* < mailing-list-file". *checkaddr* announces each address on a separate line and follows the address with its status (normally "OK"). *checkaddr* uses *submit*(ADM) to do the address verification.

If the **-w** option is given, *checkaddr* causes *submit* to generate a detailed submission tracing. This can sometimes be useful to help find problems in alias files or mailing lists.

See Also

submit(ADM)

checkque

MMDF queue status report generator

Syntax

checkque [**-fpsz**] [**-t**age[m]] [**-c** channel channel ...]

Description

checkque reports on the amount of mail waiting in the MMDF distribution queue. It indicates the total number of messages and the size of the queue directory. It then lists the number of messages waiting for each transmission channel.

The **-c** option allows one or more channel names to be specified. If present, *checkque* restricts it's report to the named channels.

The **-f** option causes *checkque* to print the name of the oldest queued message for each channel. **-p** causes only channels with "problems" to be listed. Problems are defined as channels which have mail waiting for over some "problem threshold." The default "problem threshold" is 24 hours. The **-t** option is used to change the "problem threshold." A number of hours (or minutes, if "m" is appended) should appear without a space after the **-t**. -s forces an abbreviated summary listing instead of the normal multi-line report. **-z** causes channels with no messages queued to be skipped in the report.

Since the mail queue usually is protected from access by any uid, except MMDF, *checkque* should be run under root or MMDF uid. It should not be made *setuid*() to mmdf unless you want to allow non-staff members to see the queue status.

Most configurations will have only two channels. One is for local delivery and the second is for off-machine relaying, such as by calling out or by being called up, or by attaching to ArpaNet hosts. Local delivery usually happens at the time of submission, so it is rare that any mail is waiting in it. Mail in other outbound queues is processed by *deliver* according to your site parameters, either by running *deliver* as a background daemon or by periodically firing it up via *cron.*

Files

quedfldir[]/addr
quedfldir[]/msg
quedfldir[]/q.*

*phase-directory/channel/**

See Also

phs(S), deliver(ADM)

Author

Dave Crocker, Dept. of E.E., Univ. of Delaware

checkup

Report on MMDF problems

Syntax

/usr/mmdf/bin/checkup [-p -v[*digit*]]

Description

The *checkup* command is used to check aspects of the MMDF system configuration. Normally, *checkup* reports on all problems that are encountered, including correct states. Displayed problems are prefixed by two asterisks (**); information that is advisory is enclosed in square brackets ([]).

The two optional flags to *checkup* specify how much information is displayed. The **-p** option reports only problems detected by *checkup*. This is useful for day-to-day checking of the system, such as mailing the output to the postmaster alias.

The **-v** flag takes an optional *digit* which ranges from 1 (the same as the **-p**) option, to level 7 which displays all information.

Some of the displayed information, such as that about permissions modes varies by site conventions and may not have widespread significance. In particular it is common for sites to allow group read, write, or execute on files that *checkup* expects to be protected more carefully. Use of group permissions can greatly ease administration efforts for system managers without compromising security. Warnings regarding ''others'' permissions should be examined.

chg_audit

enables and disable auditing for the next session

Syntax

/tcb/lib/chg_audit [on]

Description

chg_audit enables and disable auditing for the next session (next reboot). It edits the **/etc/inittab** and **/etc/conf/cf.d/init.base** file to add or remove the audit startup command when the system is rebooted. The command is normally invoked by the *auditsh*(ADM).

If **on** is specified, then auditing is enabled. If no argument is given, then the audit lines are removed from the **inittab** files.

Files

/etc/inittab
/etc/conf/cf.d/init.base

See Also

auditsh(ADM), sysadmsh(ADM)

Value Added

chg_audit is an extension of AT&T System V provided in Altos UNIX System V.

chroot

changes root directory for command

Syntax

chroot newroot command

Description

The given command is executed relative to the new root. The meaning of any initial slashes (/) in pathnames is changed for a command and any of its children to *newroot*. Furthermore, the initial working directory is *newroot*.

Notice that:

chroot newroot command >x

creates the file **x** relative to the original root, not the new one.

This command is restricted to the super-user.

The new root pathname 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. Note that it is not possible to change directories to what was formerly the parent of the new root directory; i.e., the *chroot* command supports the new root as an absolute root for the duration of the *command*. This means that "/.." is always equivalent to "/".

See Also

chdir(S)

Notes

Exercise extreme caution when referencing special files in the new root file system.

command must be under *newroot* or *command* is reported:
command: not found

Standards Conformance

chroot is conformant with:

AT&T SVID Issue 2, Select Code 307-127;
and The X/Open Portability Guide II of January 1987.

cleanque

send warnings and return expired mail

Syntax

cleanque [-w]

Description

cleanque removes extraneous files from the **tmp** and **msg** subdirectories of the MMDF "home queue" directory. It also sends warnings for mail which has not been fully delivered after "warntime" hours following submission. Finally, it returns mail which has not been fully delivered after "failtime" hours after submission. "Warntime" and "failtime" are defined in the MMDF *mmdftailor*(F) file.

Generally, *cleanque* should be run by *cron,* once a day, but may be run at any time to free up space.

The optional argument, **-w**, can be used if you are running *cleanque* manually and want to see what the program is doing.

See Also

queue(F), deliver(ADM)

Notes

cleanque does not currently remove extraneous files from the individual queues (**q.*** subdirectories).

cleantmp

remove temporary files in directories specified

Syntax

/usr/lib/cleantmp

Description

cleantmp removes temporary files in directories specified in **/etc/default/cleantmp** under the variable TMPDIRS. By default, **/tmp** and **/usr/tmp** are examined. Users can add to the list of directories, separating each directory with a space. Files in these directories which are not accessed within the last n days will be removed, where n is the number of days specified under the variable FILEAGING in **/etc/default/cleantmp**. By default, FILEAGING is 7. Users can change the number of days for FILEAGING. **/usr/lib/cleantmp** is run as a cron job every day at 3:00a.m. Refer to **/usr/spool/cron/crontabs/root** on the system. The super user can edit this file to change the frequency and time at which **/usr/lib/cleantmp** is run. If the directories specified do not exist or if they are mount points and the file system are not mounted, cleantmp will send mail to root saying that the directory does not exist.

The format of **/etc/default/cleantmp** is as follows:

```
FILEAGING=7
TMPDIRS=/tmp /usr/tmp
```

Files

/etc/default/cleantmp

See Also

rc2(ADM)

Value Added

cleantmp is an extension of AT&T System V provided in Altos UNIX System V.

clri

clears inode

Syntax

/etc/clri filesystem i-number ...

Description

clri writes zeros on the 64 bytes occupied by the inode numbered *i-number*. *Filesystem* must be a special filename referring to a device containing a file system. After *clri* is executed, any blocks in the affected file will show up as ''missing'' if the file system is checked with *fsck*(ADM). Use *clri* only in emergencies and exercise extreme care.

Read and write permission is required on the specified *filesystem* device. The inode becomes allocatable.

The primary purpose of this routine is to remove a file which, for some reason, does not appear in a directory. If you use *clri* to destroy an inode which does appear in a directory, track down the entry and remove it. Otherwise, when the inode is reallocated to some new file, the old entry will still point to this file. At that point removing the old entry will destroy the new file. The new entry will again point to an unallocated inode, so the whole cycle is likely to be repeated again and again.

See Also

fsck(ADM), ncheck(ADM)

Notes

If the file is open, *clri* is likely to be ineffective.

This utility does not work on DOS filesystems.

configure

kernel configuration program

Syntax

/etc/conf/cf.d/configure [options] [resource=value ...]

Description

The *configure* program determines and alters different kernel resources. For end users, using *configure* is easier than modifying the system configuration files directly. For device driver writers, *configure* avoids the difficulties of editing configuration files that have already been edited by an earlier driver configuration script.

You must move to the **/etc/conf/cf.d** to execute *configure.*

Resources are modified interactively or with command-line arguments. Adding or deleting device driver components requires the command-line options.

The next paragraphs discuss how to use *configure* interactively. Command-line options are discussed in the ''Options'' section.

Before using *configure*, to modify the system configuration files, use the following command to make a backup copy of the kernel.

```
cp /unix /unix.old
```

Interactive Usage

configure functions interactively when no options (including resource=value) are given or when **-f** is the only option specified on the command line.

When you invoke *configure* interactively, you first see a category menu that looks something like this:

```
1.  Disk and Buffers
2.  Character Buffers
3.  Files, Inodes, and Filesystems
4.  Processes, Memory Management and Swapping
5.  Clock
6.  MultiScreens
7.  Message Queues
8.  Semaphores
9.  Shared Data
10. System Name
11. Streams Data
12. Event Queues and Devices
13. Hardware Dependent Parameters
14. Remote File sharing Parameters

Select a parameter category to reconfigure
by typing a number from 1 to 14, or type 'q' to quit:
```

To choose a category, enter its number (e.g., "1" for "Disk Buffers"), then press ⟨Return⟩.

Each category contains a number of configurable resources. Each resource is presented by displaying its true name, a short description, and its current value. For example, for the "Disk Buffers" category you might see:

```
NBUF: total disk buffers.
Currently determined at system start up:
NSABUF: system-addressable (near) disk buffers.
Currently 10:
NHBUF: hash buffers (for disk block sorting).
Currently 128:
```

To keep the current value, simply press ⟨Return⟩. Otherwise, enter an appropriate value for the resource, then press ⟨Return⟩. *configure* checks each value to make sure that it is within an appropriate range. If not, *configure* warns you that the value is inappropriate and will confirm that you want to override the recommended value.

To exit from *configure*, enter **q** at the category menu prompt. If any changes are made, *configure* asks if it should update the configuration files with the changes. To keep the old configuration values, enter **n** at this prompt, and no changes are made. Otherwise, enter **y** and *configure* updates the required system configuration files. After *configure* has completed, the kernel is ready for linking.

To link the kernel, enter:

/etc/conf/cf.d/link_unix

Linking may take a few minutes. After the kernel is linked, enter the following command to reboot the system to run the new kernel:

/etc/shutdown

Next, you see the boot prompt:

Boot
:

Press ⟨Return⟩. The system is now running the new kernel.

Options

The command line options are designed for writers of driver-installation shell scripts. You can configure drivers, remove driver definitions from the configuration files, and modify some driver attributes, all from the command line. There are also options for querying the current driver configuration.

configure uses the following options:

-a [*func1 func2* ...]
-b
-c
-d [*func1 func2* ...]
-f *master_file* [*dfile*]
-g *dev_name handler* | *dev_name*
-h *dev_name*
-j [*prefix*] [**NEXTMAJOR**]
-l *priority_level*
-m *major_dev_number*
-o
-s
-t
-v *interrupt_vector* [*interrupt_vector2...*]
-w
-x
-y *resource*
-A *address address*
-C *channel*
-D
-G
-H
-I *address address*
-J *address address*
-M *maximum minimum*

-O
-P
-R
-S
-T *interrupt_scheme*
-U *number_of_subdevices*
-V *interrupt_vector*
-Y
-Z

-m, **-b**, and **-c**
These options are used to define which driver is being referenced. Following **-m** must be the major device number of the driver. If you are configuring a block driver, **-b** must appear; if you are configuring a character driver, **-c** must appear. Both are used when configuring a driver with both kinds of interfaces.

-s When adding or deleting a streams module, use this option with the **-h** option and instead of **-m**, **-b**, and **-c**. For a streams driver, use it with **-m** and **-c**.

-a and **-d**
Each option is followed by a list of functions to add or delete, respectively. These are the names of the functions that appear within **bdevsw[ ]** or **cdevsw[ ]**, as appropriate, plus the names of the initialization, clock poll, halt, and interrupt routines, if present, plus the name of the tty structure pointer. *configure* enforces the rules that all of a driver's routines must have a common prefix, and that the prefix be 2-4 characters long.

-h This option is used to give the driver or streams module name when the name is different from the prefix or when no prefix is specified as in the case of the streams module. The name can be 1-8 characters long.

-j When followed by a *prefix* used by a driver, the major device number is displayed. When followed by **NEXTMAJOR**, the smallest major device number is displayed.

-v This option modifies the system notion of the vectors on which this device can interrupt.

-l This sets the interrupt priority level of the device, which is almost always the same as the type of *spl*( ) call used: a driver that interlocks using *spl5*( ) almost always has an interrupt priority level of 5. Use of this option in new drivers is not recommended.

-f Much of the configuration data is maintained in two files, whose default names are **mdevice** and **mtune**. The **-f** option can be used to specify alternate names. Note that if **-f** is the only option present, the program is still interactive.

-w When specifying a parameter value, this option suppresses warning messages.

-o This is the override flag. When invoked non-interactively, this option overrides the minimum and maximum values that are otherwise enforced. No warnings are given. This option has no effect on interactive commands.

-x This dumps all the resource prompts known to *configure*. These reveal the name, description, and current value of each parameter capable of being reconfigured. Category prompts are not dumped.

-y The **-y** option displays the current value of the requested parameter.

-t This option displays nothing (except possibly error messages). However, it has a return value of 1 if a driver corresponding to the given combination of **-m**, **-b**, **-c** and options is already configured, and returns 0 if no such driver is present.

-g This option is used to add or remove graphics input (GIN) device handlers. Devices such as mice, bitpads, and keyboards may have handlers to turn their input data into "events." The **-g** flag may be given one argument that is interpreted as a device name. That GIN device is removed from the configuration files. If the **-g** flag has two arguments, the second is a handler for that device, and the device is added to the files. If it was already present, its handler is updated and the user is informed. Multiple devices may be added or removed by specifying **-g** multiple times.

-A This option, followed by two values that are taken to be hexadecimal I/O addresses, returns the name of the device with the I/O address conflict.

-C Followed by an integer, this option used with **-a** indicates the DMA channel that the device uses. The default is not to use DMA.

-D This option used with the **-a** option adds to the device driver the characteristic that the driver can share its DMA channel; **-D** used with the **-d** option deletes this characteristic. The default is not to share.

-G This option with **-a** adds the *G* characteristic to the driver; **-G** with **-d** deletes the *G* characteristic. This characteristic indicates whether or not the device uses an interrupt, even though an interrupt is specified in the **sdevice** entry. This is used when you want to associate a device to a specific device group. The default is not to set this characteristic.

-H This option with **-a** or **-d** adds or deletes the characteristic that the driver supports hardware that distinguishes it from those that are entirely software (pseudo devices). The default is to set this characteristic.

-I This option is followed by two values that are the hexadecimal start and end I/O addresses. The default values are zero.

-J The option is followed by two values that are the hexadecimal start and end controller memory addresses. The default values are zero.

-M
This option followed by two integers states the maximum and minimum number of devices that can be specified in the **sdevice** file. The default is a maximum of 1 and a minimum of 0.

-O This option with **-a** or**-d** indicates whether or not the IOA range of the device can overlap that of another device. The default is no.

-P When used with **-a** or **-d**, adds or deletes an ignore "I" flag in the device *mdevice* entry. The "I" flag allows the configuration build utilities to ignore a devices **pack.d** directory (useful to the mpt/spt) driver.

-R This option with **-a** or**-d** indicates whether or not the driver is required in the kernel all the time. The default is yes.

-S This option with **-a** or**-d** indicates whether or not the driver has one **sdevice** entry only. The default is no.

-T This option, when followed by an argument, states the type of interrupt scheme the device uses. The possible arguments are:

0 The device does not require an interrupt line.

1 The device requires an interrupt line. If the device supports more than one controller, each controller requires a separate interrupt.

2 The device requires an interrupt line. If the device supports more than one controller, the controllers share the same interrupt.

3 The device requires an interrupt line. If the device supports more than one controller, the controllers share the same interrupt. Multiple device drivers having the same interrupt priority level can share this interrupt.

The default is 0.

-U This option, when followed by an integer, encodes a device-dependent numeric value in the **sdevice** file to indicate the number of subdevices on a controller or a pseudo device. The integer must be a value that lies within the maximum and minimum number of devices specified in the **mdevice** file. The default is 1.

-V This option, followed by a vector value, returns the name of the device with the vector conflict.

-Y This option with **-a** or-**d** indicates whether or not to configure a driver into the kernel. Specifying **-a** puts a "Y" in the configuration field of the driver's **sdevice** entry; specifying **-d** puts an "N" in this field. The default is to put a "Y".

-Z This option indicates that a device can have more than one entry in the **mdevice** file. The SCSI driver is an example of a driver that needs this feature. The option is usually used when adding a new entry or deleting a particular entry in the **mdevice** file. Using **-d** with **-Z** removes only the **mdevice** entry. Using **-d** without **-Z** removes the **mdevice** entry and the **sdevice** entry.

Setting Command-Line Parameters

Any number of arguments can be given on the command line of the form *resource=value*. These arguments can be given at the same time as an add or delete driver request, but must follow all the driver-configuration arguments on the command line.

If one or more instances of *resource=value* are the only arguments on the command line, the changes are made non-interactively. If the values given are outside the permissible range for a parameter, no action is taken unless the **-o** option is included to override them.

Some resources have values that are character strings. In this case, their values must be enclosed within the characters \" . The quotes are syntactically necessary for them to be used as C-language strings, and the backslashes protect the quotes from being removed by the shell.

Examples

Print out the current value of NCLIST:

configure -y NCLIST

Return 1 if character major device 7 and vector 3 are already configured:

configure -t -v 3 -m 7 -c

Add a clock-time polling and initialization routine to the already configured "foo" driver, a hypothetical character driver at major device #17:

configure -a foopoll fooinit -c -m 17

Delete the "foo" driver:

configure -m 17 -d -c

Add a new "hypo" driver, a block driver with a character interface. It absorbs 3 different interrupt vectors, at priority 6:

**configure -a hypoopen hypoclose hyporead hypowrite hypoioctl \
hypostrategy hypoprint hypointr -b -c -l 6 -v 17 42 49 -m 10**

Add a new streams module with prefix "grb" and name "garble":

configure -s -a grbinit -h garble

Files

/etc/conf/cf.d/mdevice
/etc/conf/cf.d/sdevice
/etc/conf/cf.d/mtune
/etc/conf/cf.d/stune
/etc/conf/cf.d/mevent
/etc/conf/cf.d/sevent

See Also

idconfig(ADM), link_unix(ADM), majorsinuse(ADM), routines(ADM), vectorsinuse(ADM), mdevice(F), mtune(F), sdevice(F), stune(F), event(M), "Tuning System Performance" in the *System Administrator's Guide*

Value Added

configure is an extension of AT&T System V provided in Altos UNIX System V.

consoleprint

print /usr/adm/messages or any file to a serial printer attached to the printer port of a serial console

Syntax

consoleprint [file]

Description

consoleprint prints the file **/usr/adm/messages** to a printer attached to the printer port of a serial console. If a filename is specified, it is printed instead. *consoleprint* is normally run by a system administrator to get a hardcopy version of the system console messages.

This command uses the file **/etc/termcap**.

Files

/etc/termcap

See Also

lprint(C)

Notes

The only terminals currently supported with entries in /etc/termcap are the Tandy DT-100 and DT-1, and the Hewlett-Packard HP-92.

Terminal communications parameters (such as baud rate and parity) must be set up on the terminal by the user.

Value Added

consoleprint is an extension of AT&T System V provided in Altos UNIX System V.

crash

examine system images

Syntax

/etc/crash [**-r**] [**-d** dumpfile] [**-n** namelist] [**-o** offset]
[**-w** outputfile]

Description

The *crash* command is used to examine the system memory image of a live or a crashed system by formatting and printing control structures, tables, and other information. Command line arguments to *crash* are *dumpfile, namelist, offset*, and *outputfile.*

Dumpfile is the file containing the system memory image. The default *dumpfile* is */dev/mem.*

If the **-r** option is used, then *dumpfile* is assumed to contain a system memory image as dumped after a UPS power failure shutsave operation. The information stored in the restart header is then displayed. This option causes *crash* to display restart information only, and will not enter a normal session. Typically, the **-r** option is used in conjunction with **-d /dev/restart**.

The text file *namelist* contains the symbol table information needed for symbolic access to the system memory image to be examined. The default *namelist* is */unix.* If a system image from another machine is to be examined, the corresponding text file must be copied from that machine. The offset is used to specify the starting location of the memory dump relative to the beginning of *dumpfile.* The default is 0 (zero). If the system memory image is a UPS shutsave dump (i.e., in **/dev/restart**), then an offset of 1024 must be used, since in this case the actual memory image begins at offset 1024, after the restart header.

When the *crash* command is invoked, a session is initiated. The output from a *crash* session is directed to *outputfile.* The default *outputfile* is the standard output.

Input during a *crash* session is of the form:

function [argument ...]

where *function* is one of the *crash* functions described in the Functions section of this manual page, and *arguments* are qualifying data that indicate which items of the system image are to be printed.

The default for process-related items is the current process for a running system and the process that was running at the time of the crash for a crashed system. If the contents of a table are being dumped, the default is all active table entries.

The following function options are available to *crash* functions wherever they are semantically valid.

-e Display every entry in a table.

-f Display the full structure.

-p Interpret all address arguments in the command line as *physical* addresses.

-s *process*
Specify a process slot other than the default.

-w *file*
Redirect the output of a function to *file*.

Note that if the **-p** option is used, all address and symbol arguments explicitly entered on the command line will be interpreted as physical addresses. If they are not physical addresses, results will be inconsistent.

The functions *mode*, *defproc*, and *redirect* correspond to the function options **-p**, **-s**, and **-w**. The *mode* function may be used to set the address translation mode to physical or virtual for all subsequently entered functions; *defproc* sets the value of the process slot argument for subsequent functions; and *redirect* redirects all subsequent output.

Output from *crash* functions may be piped to another program in the following way:

```
function [ argument ... ] ! shell_command
```

For example,

```
mount ! grep rw
```

will write all mount table entries with an *rw* flag to the standard output. The redirection option (**-w**) cannot be used with this feature.

Depending on the context of the function, numeric arguments will be assumed to be in a specific radix. Counts are assumed to be decimal. Addresses are always hexadecimal. Table slot arguments are always decimal. Table slot arguments larger than the size of the function table will not be interpreted correctly. Use the *findslot* command to translate from an address to a table slot number. Default bases on all arguments may be overridden. The C conventions for designating the bases of numbers are recognized. A number that is usually interpreted as decimal will be interpreted as hexadecimal if it is preceded by **0x** and as octal if it is preceded by **0**. Decimal override is designated by **0d**, and binary by **0b**.

Aliases for functions may be any uniquely identifiable initial substring of the function name. Traditional aliases of one letter, such as **p** for *proc*, remain valid.

Many functions accept different forms of entry for the same argument. Requests for table information will accept a table entry number or a range. A range of slot numbers may be specified in the form *a-b* where *a* and *b* are decimal numbers. An expression consists of two operands and an operator. An operand may be an address, a symbol, or a number; the operator may be +, -, *, /, &, or | . An operand which is a number should be preceded by a radix prefix if it is not a decimal number (**0** for octal, **0x** for hexadecimal, **0b** for binary). The expression must be enclosed in parentheses ( ). Other functions will accept any of these argument forms that are meaningful.

Two abbreviated arguments to *crash* functions are used throughout. Both accept data entered in several forms. They may be expanded into the following:

table_entry = table entry| range

start_addr = address| symbol| expression

Functions

? [**-w** file]
List available functions.

!cmd
Escape to the shell to execute a command.

adv [**-e**] [**-w** file] [[**-p**] table_entry ...]
Print the advertised table.

base [**-w** file] number ...
Print *number* in binary, octal, decimal, and hexadecimal. A number in a radix other than decimal should be preceded by a prefix that indicates its radix as follows: **0x**, hexadecimal; **0**, octal; and **0b**, binary.

buffer [**-w** file] [-format] bufferslot

or

buffer [**-w** file] [-format] [**-p**] start_addr
Alias: **b**.
Print the contents of a buffer in the designated format. The following format designations are recognized: **-b**, byte: **-c**, character; **-d**, decimal; **-x**, hexadecimal; **-o**, octal; **-r**, directory; and **-i**, inode. If no format is given, the previous format is used. The default format at the beginning of a *crash* session is hexadecimal.

bufhdr [**-f**] [**-w** file] [[**-p**] table_entry ...]
Alias: **buf**.
Print system buffer headers.

callout [**-w** file]
Alias: **c**.
Print the callout table.

dballoc [**-w** file] [class ...]
Print the dballoc table. If a class is entered, only data block allocation information for that class will be printed.

dbfree [**-w** file] [class ...]
Print free streams data block headers. If a class is entered, only data block headers for the class specified will be printed.

dblock [**-e**] [**-w** file] [**-c** class ...]

or

dblock [**-e**] [**-w** file] [[**-p**] table_entry ...]
Print allocated streams data block headers. If the class option (**-c**) is used, only data block headers for the class specified will be printed.

defproc [**-w** file] [**-c**]

or

defproc [**-w** file] [slot]
Set the value of the process slot argument. The process slot argument may be set to the current slot number (**-c**) or the slot number may be specified. If no argument is entered, the value of the previously set slot number is printed. At the start of a *crash* session, the process slot is set to the current process.

dis [**-w** file] [**-a**] start_addr [count]
Disassemble from the start address for *count* instructions. The default count is 1. The absolute option (**-a**) specifies a non-symbolic disassembly.

ds [**-w** file] virtual_address ...
Print the data symbol whose address is closest to, but not greater than, the address entered.

file [**-e**] [**-w** file] [[**-p**] table_entry ...]
Alias: **f**.
Print the file table.

findaddr [**-w** file] table slot
Print the address of *slot* in *table*. Only tables available to the *size* function are available to *findaddr*.

findslot [**-w** file] virtual_address ...
Print the table, entry slot number, and offset for the address entered. Only tables available to the *size* function are available to *findslot*.

fs [**-w** file] [[**-p**] table_entry ...]
Print the file system information table.

gdp [**-e**] [**-f**] [**-w** file] [[**-p**] table_entry ...]
Print the gift descriptor protocol table.

gdt [**-e**] [**-w** file] [[**-p**] table_entry ...]
Print the global descriptor table.

help [**-w** file] function ...
Print a description of the named function, including syntax and aliases.

idt [**-e**] [**-w** file] [[**-p**] table_entry ...]
Print the interrupt descriptor table.

inode [**-e**] [**-f**] [**-w** file] [[**-p**] table_entry ...]
Alias: **i**.
Print the inode table, including file system switch information.

kfp [**-w** file] [value]
Print the frame pointer for the start of a kernel stack trace. If the value argument is supplied, the kfp is set to that value.

lck [**-e**] [**-w** file] [[**-p**] table_entry ...]
Alias: **l**.
Print record-locking information. If the **-e** option is used or table address arguments are given, the record lock list is printed. If no argument is entered, information on locks relative to inodes is printed.

ldt [**-e**] [**-w** file] [-s process] [[**-p**] table_entry ...]
Print the local descriptor table for the given process, or for the current process if none is given.

linkblk [**-e**] [**-w** file] [[**-p**] table_entry ...]
Print the linkblk table.

map [**-w** file] mapname ...
Print the map structure of *mapname*.

mbfree [**-w** file]
Print free streams message block headers.

mblock [**-e**] [**-w** filename] [[**-p**] table_entry ...]
Print allocated streams message block headers.

mode [**-w** file] [mode]
Set address translation of arguments to virtual (**v**) or physical (**p**) mode. If no mode argument is given, the current mode is printed. At the start of a *crash* session, the mode is virtual.

mount [**-e**] [**-w** file] [[**-p**] table_entry ...]
Alias: **m**.
Print the mount table.

nm [**-w** file] symbol ...
Print value and type for the given symbol.

od [**-p**] [**-w** file] [**-**format] [**-**mode] [**-s** process] start_addr [count]
Alias: **rd**.
Print *count* values starting at the start address in one of the following formats: character (**-c**), decimal (**-d**), hexadecimal (**-x**), octal (**-o**), ASCII (**-a**), or hexadecimal/character (**-h**), and one of the following modes: long (**-l**), short (**-t**), or byte (**-b**). The default mode for character and ASCII formats is byte; the default mode for decimal, hexadecimal, and octal formats is long. The format **-h** prints both hexadecimal and character representations of the addresses dumped; no mode needs to be specified. When format or mode is omitted, the previous value is used. At the start of a *crash* session, the format is hexadecimal and the mode is long. If no count is entered, 1 is assumed.

panic
Print the latest system notices, warnings, and panic messages from the limited circular buffer kept in memory.

pcb [**-w** file] [process]
Print the process control block (TSS) for the given process. If no arguments are given, the active TSS for the current process is printed.

pdt [**-e**] [**-w** file] [**-s** process] [**-p**] start_addr [count]
The page descriptor table of the designated memory *section* and *segment* is printed. Alternatively, the page descriptor table starting at the start address for *count* entries is printed. If no count is entered, 1 is assumed.

pfdat [**-e**] [**-w** file] [[**-p**] table_entry ...]
Print the pfdata table.

proc [-e] [**-f**] [**-w** file] [[**-p**] table_entry ... #procid ...]

or

proc [**-f**] [**-w** file] [**-r**]
Alias: **p**.
Print the process table. Process table information may be specified in two ways. First, any mixture of table entries and process ids may be entered. Each process id must be preceded by a #. Alternatively, process table information for executable processes may be specified with the executable option (**-r**). The full option (**-f**) details most of the information in the process table as well as the region table for that process.

qrun [**-w** file]
Print the list of scheduled streams queues.

queue [**-e**] [**-w** file] [[**-p**] table_entry ...]
Print streams queues.

quit
Alias: **q**.
Terminate the *crash* session.

rcvd [**-e**] [**-f**] [**-w** file] [[**-p**] table_entry ...]
Print the receive descriptor table.

redirect [**-w** file] [**-c**]

or

redirect [**-w** file] [file]
Used with a file name, redirects output of a *crash* session to the named file. If no argument is given, the file name to which output is being redirected is printed. Alternatively, the close option (**-c**) closes the previously set file and redirects output to the standard output.

region [-e] [**-w** file] [[**-p**] table_entry ...]
Print the region table.

sdt [**-e**] [**-w** file] [-s process] section

or

sdt [**-e**] [**-w** file] [-s process] [**-p**] start_addr [count]
The segment descriptor table for the current process is printed.

search [**-p**] [**-w** file] [**-m** mask] [**-s** process] pattern start_addr count
Print the long words in memory that match *pattern*, beginning at the start address for *count* long words. The mask is anded (&) with each memory word and the result compared against the pattern. The mask defaults to 0xffffffff.

size [**-w** file] [**-x**] [structure_name ...]
Print the size of the designated structure. The (**-x**) option prints the size in hexadecimal. If no argument is given, a list of the structure names for which sizes are available is printed.

sndd [**-e**] [**-f**] [**-w** file] [[**-p**] table_entry ...]
Print the send descriptor table.

srmount [**-e**] [**-w** file] [[**-p**] table_entry ...]
Print the server mount table.

stack [**-w** file] [process]
Alias: **s**.
Dump stack. If no arguments are entered, the kernel stack for the current process is printed. The interrupt stack and the stack for the current process are not available on a running system.

stat [**-w** file]
Print system statistics.

stream [**-e**] [**-f**] [**-w** file] [[**-p**] table_entry ...]
Print the streams table.

strstat [**-w** file]
Print streams statistics.

trace [**-w** file] [**-r**] [process]
Alias: **t**.
Print kernel stack trace. The kfp value is used with the **-r** option.

ts [**-w** file] virtual_address ...
Print closest text symbol to the designated address.

tty [**-e**] [**-f**] [**-w** file] [**-t** type [[**-p**] table_entry ...]]
Valid types: **co**, **c1**, **c2** (console, com1, com2).
Print the tty table. If no arguments are given, the tty table for the console is printed. If the **-t** option is used, the table for the single tty type specified is printed. If no argument follows the type option, all entries in the table are printed. A single tty entry may be specified from the start address.

user [**-f**] [**-w** file] [process]
Alias: **u**.
Print the ublock for the designated process.

var [**-w** file]
Alias: **v**.
Print the tunable system parameters.

vtop [**-w** file] [**-s** process] start_addr ...
Print the physical address translation of the virtual start address.

Files

/dev/mem	system image of currently running system
/dev/restart	system image of a UPS shutsave dump

See Also

upsconfig(ADM)

custom

installs specific portions of the UNIX System

Syntax

custom [**-od**] [**-irla** [package]] [**-m** device] [**-f** [file]]

Description

With *custom* you can create a custom installation by selectively installing or deleting portions of the UNIX system. *custom* is executable only by the super-user and is either interactive or can be invoked from the command line with several options.

Files are extracted or deleted in *packages*. A package is a collection of individual files.

You can also install additional *sets*. You can list the available *packages* by using the *custom* command as described next.

Usage

To use *custom* interactively, enter:

custom

The *custom* main menu appears with the following options:

Install
Allows a product or system to be added.

A window is first opened to select the system set or product. When a system or product is selected, you are given the choice of adding the ''Entire Product'', ''Packages'' or ''Files''. When ''Entire Product'' is chosen, *custom* calculates which installation volumes (distribution media) are needed, then prompts for the correct volume numbers.

If ''Packages'' is chosen, a list of all available packages in the currently selected set is displayed. Each line describes the package name, whether the package is fully installed, not installed or partially installed, the size of the package (in 512 byte blocks), and a one line description of the package contents.

Multiple packages can be specified by marking them with the space bar. The selected packages will appear with asterisks. When executed, *custom* will prompt for insertion of the necessary volumes. (You cannot use *custom* to install the entire RTS package if that package is already partially installed. If this situation comes up, use *fixperm*(ADM) to determine which files are missing, and then use *custom* to install each file individually.)

If "Files" is chosen, you are prompted to select the package and then the file names. *custom* then prompts for volumes.

Remove
Deletes the correct files in the specified package/product. Select the product or package to be deleted just as you select a product or package to install.

List
Lists all files in the specified package or all packages in a product set.

Quit
Leaves *custom*.

Options

Three arguments are required for a completely non-interactive use of *custom*:

A set identifier
(-**o** or -**d**)

A command
(-**i**, -**r**, -**l**,-**f**, or -**a**)

And either one or more package names, or a file name

If any information is missing from the command line, *custom* prompts for the missing data.

Only one of -**o**, or -**d** may be specified. These stand for:

-**o** Operating System

-**d** Development System

Only one of **-i**, **-r**, **-l**, **-f**, or **-a** may be specified, followed by an argument of the appropriate type (one or more package names, or a file name). These options perform the following:

-i Install the specified package(s)

-r Remove the specified package(s)

-l List the files in the specified package(s).

-f Install the specified file.

-a Add a new product

The **-m** flag allows the media device to be specified. The default is /dev/install (which is always the 0 device, as in /dev/fd0). This is very useful if the system has a 5.25-inch drive on /dev/fd0 and a 3.5-inch floppy on /dev/fd1, and it is necessary to install 3.5-inch media. For example:

custom -m /dev/rfd196ds9

this will override the default device and use the one supplied with the **-m** flag.

Files

/etc/perms/*

See Also

fixperm(ADM), df(C), du(C), xinstall(ADM)

Notes

If you upgrade any part of your system, *custom* detects if you have a different release and prompts you to insert the floppy volume that updates the custom data files. Likewise, if you insert an invalid product or a volume out of order, you will be prompted to reinsert the correct volume.

Upon installation of the operating system, the RTS package is always entirely installed.

Value Added

custom is an extension to AT&T System V provided in Altos UNIX System V.

dbmbuild

builds the MMDF hashed database of alias and routing information

Syntax

/usr/mmdf/table/dbmbuild [-nvdk]] [*database* [*table* ...]]

Description

dbmbuild reads the tables specified in the MMDF tailor file into a hashed database for use in quickly verifying addresses and efficiently assigning channels to submitted messages. Whenever you change MMDF alias or routing information in any way, you must rebuild the hashed database by logging in as *mmdf* and running *dbmbuild* from the */usr/mmdf/table* directory.

If no database file is specified, the default database **mmdfdbm** is used. If no table files are specified, all tables listed in the tailor file are used. In particular, three tables are read for each channel definition: the list of authorized sources, the list of authorized destinations, and the table of names/aliases for that channel. Also, the remaining tables (MTBL and MDMN) are read.

The options are:

n Create a new database. If this option is omitted, *dbmbuild* updates an existing database. If no options at all are specified, **-n** is assumed; however, if you give any options (even **-v**), you must specify the **-n** option if you want to create a new database.

v Run in verbose mode, displaying information during table processing.

d Run in debug mode, reporting everything that happens.

k Keep going. If a file is mentioned that does not exist, ignore it. This option might be an appropriate default at some sites.

Appropriate locks are placed on the database so that *dbmbuild* can safely be run while MMDF is in operation.

Files

/usr/mmdf/mmdftailor	
/usr/mmdf/table/alias.list	
/usr/mmdf/table/alias.user	
/usr/mmdf/table/*.chn	
/usr/mmdf/table/*.dom	
$(tbldbm).dir	database directory
$(tbldbm).pag	database pages
$(tbldbm).lck	database locking file
$(tbldfldir)/*	various tables that form the database

See Also

tables(F), mmdftailor(F), dbm(S), ''Setting Up Electronic Mail'' in the *System Administrator's Guide*

dcopy

copy UNIX filesystems for optimal access time

Syntax

/etc/dcopy [-sX] [-an] [-d] [-v] [-ffsize[:isize]] inputfs outputfs

Description

The *dcopy* command copies filesystem *inputfs* to *outputfs*. *Inputfs* is the device file for the existing file system; *outputfs* is the device file to hold the reorganized result. This utility is for UNIX fileystems only. For the most effective optimization, *inputfs* should be the raw device and *outputfs* should be the block device. Both *inputfs* and *outputfs* should be unmounted file systems.

With no options, *dcopy* copies files from *inputfs* compressing directories by removing vacant entries, and spacing consecutive blocks in a file by the optimal rotational gap. The possible options are:

-s*X* supply device information for creating an optimal organization of blocks in a file. The forms of *X* are the same as the **-s** option of *fsck*(ADM).

-a*n* place the files not accessed in *n* days after the free blocks of the destination file system (default for *n* is 7). If no *n* is specified, then no movement occurs.

-d leave order of directory entries as is (default is to move sub-directories to the beginning of directories).

-v currently reports how many files were processed, and how big the source and destination freelists are.

-f*fsize* [**:***isize*]
specify the *outputfs* file system and inode list sizes (in blocks). If the option (or **:***isize*) is not given, the values from the *inputfs* are used.

dcopy catches interrupts and quits, and reports on its progress. To terminate *dcopy* send a quit signal, followed by an interrupt or quit.

See Also

fsck(ADM), mkfs(ADM), ps(C)

deliver

MMDF mail delivery process

Syntax

deliver [-bdpsw] [-cchan,chan] [-lmins] [-thrs] [-mmaxsort] [-Llogfile] [-Tsecs] [-Vloglevel] [message1 ... messageN]

Description

The deliver program handles the management of all mail delivery under the MMDF mail system. *deliver* does not deliver mail directly, but instead calls on MMDF channels to handle actual delivery. *deliver*'s actions are guided by the MMDF tailoring file, **/usr/mmdf/mmdftailor**, and by the command line options. The program can run as either a daemon or a user-invoked program. The program may be called to process the entire mail queue or just handle some explicitly named messages. When possible, *deliver* will attempt to process messages in the order received. *deliver* also maintains a cache of host information on a per-channel basis which allows hosts which are unavailable for delivery to be skipped until available.

deliver first builds a list of channels to process, either from the command line or composed of all the non-passive channels in the system. Next, a list of messages to process is collected, either from the command line or by scanning the mail queue for for each channel. If the the number of messages in the queue for a given channel is more than *maxsort* (set in tailor file or on command line), the queue directory for that channel will be processed in the order read, without sorting by submission time. If a list of messages is given on the command line, no sorting will take place and the messages will be delivered in the order specified. The sorting keys are (in order): channel, submission time, and finally host. This causes many accesses to the messages but minimizes the invocation of channel programs.

deliver is *setuid* to the superuser to allow it to set its real and effective UID and GID to that of the MMDF user.

The following options may be used to alter *deliver*'s behavior:

-b Background mode. Causes *deliver* to run as a background daemon making periodic sweeps over the mail queues looking for undelivered mail and attempting deliver. The invoker must be the MMDF user or the superuser to use this option. *deliver* attempts delivery for all eligible messages, then sleeps, and then repeats the process. The default sleep time is 10 minutes but it can be changed (see the **-T** option below).

-cchannel1,channel2,...
Channel selection. A comma-separated list of channels to be processed.

-d Already in "quedfldir". This option will cause *deliver* to assume it is already in the mail queue and therefore it will not issue an explicit chdir(). This is useful if you wish to have deliver operate on an alternate mail queue hierarchy, mainly for testing.

-lminutes
Sets the "time-to-live" for entries in the dead-host cache. This time defaults to 2 hours. The dead host cache is used to prevent attempts to deliver to hosts that are known to be down. The "time-to-live" is given in minutes. If the number of minutes is negative, dead host caching is disabled.

-mmaxsort
Sets the sort threshold. If there are more than *maxsort* messages in a given channel's queue, then they are processed in directory order without first sorting by submission time. If -m is not specified, the value of *maxsort* is given in the tailor file by MMAXSORT.

-p Pickup only mode. Indicates that the invoker would like to pickup a passive mail channel.

-s Force linear search of the mail queue. Normally *deliver* will deliver messages in the order they were received which seldom matches the order in the directory. This option is useful if the queue gets so large that *deliver* can no longer deal with sorting the queue in a reasonable time.

-thrs
Time limiting. This option prevents *deliver* from attempting to deliver messages which have been in the queue for more than *hrs* hours. For efficiency reasons, this option only applies when the queue is being sorted. If an explicit list of messages was given on the command line, if the **-s** option is in effect, or there are more messages than the maxsort threshold (see the **-m** option), then time limiting does not occur.

-w Watch the delivery. Causes *deliver* to print informative messages on the standard output as it is attempting delivery. This option is passed onto the channel programs which also give informative messages.

-Llogfile
Sets the logfile for this deliver to the file specified. The default is to log into the file msg.log in the MMDF log directory. This option is only available to the Superuser and MMDF.

-Tseconds
Sets the sleep time between background sweeps of the mail queue. This defaults to 10 minutes.

-Vloglevel
Sets the logging level for this deliver to the level specified. The *loglevel* should be a valid mmdf logging level string such as FTR. This option is only available to the superuser and MMDF.

See Also

submit(ADM), queue(F), mmdftailor(F)

Value Added

deliver is an extension of AT&T System V provided in Altos UNIX System V.

del.vd

delete a virtual disk

Syntax

/altos/bin/del.vd [*n*]

Description

The *del.vd* utility removes a virtual disk. The system must be in singleuser mode to use this command. The virtual disk will have been previously created by *add.vd(ADM)* or *vdutil(ADM)*.

If *n* is not specified on the command line, the script prompts you for the virtual disk number. Once you reply with a correct number, the system removes the designated virtual disk.

Options

n Delete virtual disk number *n*.

See Also

add.vd(ADM), vdinfo(ADM), vddaemon(ADM), vdutil(ADM)

Note

Also refer to ''Virtual Disks'' in the *System Administrator's Guide.*

dial, uuchat

dials a modem

Syntax

/usr/lib/uucp/dial*X* ttyname telno speed
/usr/lib/uucp/dial*X* **-h** ttyname speed
/usr/lib/uucp/uuchat ttyname speed chat-script

Description

/usr/lib/uucp/dial*X* dials a modem attached to *ttyname*. (*X* is a dialer name, such as **HA1200.**) The **-h** option is used to hang up the modem.

uucico(ADM), *ct*(C), and *cu*(C) use **/usr/lib/uucp/dial***X***.** Four dialer programs are distributed. **dialHA12** is for the Hayes® Smartmodem 1200 and 1200B (and compatibles). **dialHA24** is for the Hayes® Smartmodem 2400 (and compatibles). **dialVA3450** is for the Racal-Vadic VA3450-Series dialers. **dialTBIT** is for the Telebit Trailblazer. Source for these is provided in their respective *.c* files.

uucico(ADM) invokes *dial*, with a *ttyname*, *telno* (phone number), and *speed*. *dial* attempts to dial the phone number on the specified line at the given speed. When using the **dialHA12** or **dialHA24** *speed* can be a range of baud rates. The range is specified with the form:

lowrate - *highrate*

where *lowrate* is the minimum acceptable connection baud rate and *highrate* is the maximum. The *dial* program returns the status of the attempt through the following dial return codes:

bit 0x80 = 1
The connection attempt failed.

bits 0x0f =

If bit 0x80 is a 1, then these bits are the dialer error code:

0	general or unknown error code.
1	line is being used.
2	a signal has aborted the dialer.
3	dialer arguments are invalid.
4	the phone number is invalid.
5	the baud rate is invalid or the dialer could not connect at the requested baud rate.
6	can't open the line.
7	ioctl error on the line.
8	timeout waiting for connection.
9	no dialtone was detected.
10	unused.
11	unused.
12	unused.
13	phone is busy.
14	no carrier is detected.
15	remote system did not answer.

Error codes 12-15 are used to indicate that the problem is at the remote end.

If bit 0x80 is a 0, then these bits are used to indicate the actual connection baud rate. If 0, the baud rate is the same as the baud rate used to dial the phone number or the highest baud rate if a range was specified. Otherwise, these four bits are the CBAUD bits in the **struct termio c_flag** and the **struct sgttyb sg_ispeed** and **sg_ospeed** tty ioctl structures.

You can copy and modify one of the files **/usr/lib/uucp/dialHA12.c** etc., to use a different modem. There is a makefile in **/usr/lib/uucp** which should be modified for the new dialer, and can be used to compile the new program.

If you create a *dial* program for another modem, send us the source. User generated *dial* programs will be considered for inclusion in future releases.

The *dial* program to be used on a particular line is specified in the fifth field of the entry for that line in **/usr/lib/uucp/Devices.** If there is no *dial* program of that name, then *uucico*, *ct*, and *cu* use a built-in dialer, together with the chat-script of that name in **/usr/lib/uucp/Dialers.**

dial -h is executed by *getty* when it is respawned on a line shared between dial-in and dial-out. If there is no *dial* program, then *getty* uses **/usr/lib/uucp/uuchat**, passing it the & chat-script from **/usr/lib/uucp/Dialers.**

Files

/usr/lib/uucp/Devices	
/usr/lib/uucp/dialVA3450	Racal Vadic 3450 dialer
/usr/lib/uucp/dialHA12	Hayes Smartmodem 1200/1200B dialer
/usr/lib/uucp/dialHA24	Hayes Smartmodem 2400 dialer
/usr/lib/uucp/makefile	Makefile to compile new dialer
/usr/lib/uucp/dialTBIT	Telebit Trailblazer dialer
/usr/lib/uucp/uuchat	

See Also

ct(C), cu(C), uucico(ADM), dialers(F), getty(M)

Notes

You must have the Development System installed in order to compile and install a new *dial* program.

Value Added

dial is an extension of AT&T System V provided in Altos UNIX System V.

diskusg

generate disk accounting data by user ID

Syntax

diskusg [options] [files]

Description

diskusg generates intermediate disk accounting information from data in *files*, or the standard input if omitted. *diskusg* outputs 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* (ADM)].

-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*(ADM)] which generates total accounting records that can be merged with other accounting records. *diskusg* is normally run in *dodisk* [see *acctsh*(ADM)].

Examples

The following will generate daily disk accounting information:

```
for i in /dev/dsk/0s1 /dev/dsk/0s3; do
        diskusg $i > dtmp.`basename $i` &
done
wait
diskusg -s dtmp.* | sort +0n +1 | acctdisk > disktacct
```

Files

/etc/passwd used for user ID to login name conversions

See Also

acct(ADM), acctsh(ADM), acct(F)

Standards Conformance

diskusg is conformant with:

AT&T SVID Issue 2, Select Code 307-127.

displaypkg

display installed packages

Syntax

displaypkg

Description

The *displaypkg* command will list the names of all the AT&T-style UNIX packages that were installed using the *installpkg* command.

See Also

installpkg(ADM), removepkg(ADM)

Note

This command does not work on packages installed with **custom**(ADM).

divvy

disk dividing utility

Syntax

divvy -b block_device **-c** character_device **-s** scsi_index [**-l** scsi_log_name] [**-v** virtual_drive] [**-p** physical_drive] [**-i**] [**-m**] [**-n**] [**-u**]

Description

divvy divides an *fdisk*(ADM) partition into a number of separate areas known as "divisions". A division is identified by unique major and minor device numbers and can be used for a filesystem, swap area, or for isolating bad spots on the device.

With *divvy* you can:

- Divide a disk or *fdisk* partition into separate devices.
- Create new filesystems.
- Change the size of filesystems.
- Remove filesystems.

Options

Options to *divvy* are:

-b *block_device*
Major device number of block interface.

-c *character_device*
Major device number of character interface.

-s *scsi_index*
Indicate which SCSI drive to divide, identified by the logical SCSI index number *scsi_index* (which can range from 0 to 51). Cannot use with the **-b** , **-c** , or **-p** options.

-l *scsi_log_name*
For a SCSI disk, indicate which physical disk to divide, as identified by the logical SCSI index number *scsi_log_name* . (This number should be the same as the one you used when executing **fdisk -f** *dev_name* . Cannot use with **-b** , **-c** , or **-p** options.

-v *virtual_device*
For dividing a virtual drive. (''Virtual'' here means a disk partition, not a striped or mirrored disk.)

-p *physical_drive*
For dividing one of several physical disks that share the same controller.

-i Installation only. Disk being divided will contain a **root** filesystem on division 0.

-m Disk being divided should be made into a number of mountable filesystems.

-n Installation only; non-interactive option. Disk being divided will contain the following:

root fileystem on division 0
swap on division 1
usr fileystem on division 2
scratch fileystem on division 5

Note that a **/u** filesystem may be present also, but is created only during an interactive installation.

-u Non-destructive add.

Usage

The device being divided must be a block device with a character interface. For example, to use *divvy* on a device with a block-interface major number 1 and character interface number of 1, enter:

```
divvy -b 1 -c 1
```

The **-s** and **-l** options should be used together when dividing SCSI hard disks. The **-s** option expects a logical SCSI index number, *scsi_index*, to indicate the actual drive to be installed. This number may range from 0 to 51. The **-l** option indicates the physical disk name to be added. This character should be the same as the one used when executing the **fdisk -f** *dev_name* command for this disk. The name that is chosen will have no correlation to where the disk actually resides on the system. It indicates when you added the disk. The actual disk can be determined through the major and minor device numbers, or by using the *scsinfo* (ADM) utitlity.

Note that you cannot use the **-s** and **-l** options with ESDI or ST506 drives, nor can they be used with the **-b** , **-c** , or **-p** options.

The **-v** option specifies which virtual drive to divide. The default is the active drive. Here, "virtual drive" is the same as an MS-DOS partition. Virtual drive numbers are determined with the *fdisk*(ADM) utility.

The **-p** option allows division of one of several physical disks sharing a controller. *divvy* defaults to the first physical device numbered "0." To access a second physical disk, use the **-p** 1 option.

The **-i** option is used during installation. It specifies the device being divided will contain a **root** filesystem. With this option, device nodes are created relative to the new **root**, generally a hard disk, instead of the current **root**, often an installation floppy. A root filesystem, a **/usr** filesystem, a swap area, and a recover area are created. *divvy* prompts for the size of the swap area. If the disk is large enough, then *divvy* prompts for a separate **/u** (user) filesystem. *divvy* also prompts for block-by-block control over the layout of the filesystem(s). If the root filesystem is large enough to require a scratch filesystem, (more than 40,000 blocks) then *divvy* will prompt for whether one should be created.

The **-m** option is used for initial installation on devices that will not be used as the root. It causes the user to be prompted for a number of filesystems.

The **-u** option creates device nodes for a valid division table without destroying its present contents. It will create the default name **hd***dpv*, where *d* represents the physical disk name given the **-l** option. The *p* represents the partition number specified by the **-v** option, and the *v* respresents the division number. If the **-v** option is not given, the value for *Y* defaults to 5, and the device nodes will point to the active partition.

When *divvy* is invoked from the command line, you see a main menu:

```
n[ame]        Name or rename a division.
c[reate]      Create a new file system on this division.
t[ype]        Select or change filesystem type on new filesystems.
p[revent]     Prevent a new file system from being created on this...
s[tart]       Start a division on a different block.
e[nd]         End a division on a different block.
r[estore]     Restore the original division table.

Please enter your choice or 'q' to quit:
```

To choose a command, enter the first letter of the command, then press RETURN.

The *divvy* division table might look something like this:

Name	Type	New FS	#	First Block	Last Block
root	AFS	no	0	0	69408
swap	NON FS	no	1	138818	152020
usr	AFS	no	2	69409	138817
u	AFS	no	3	152021	172020
	NOT USED	no	4	-	-
	NOT USED	no	5	-	-
recover	NON FS	no	6	172021	172030
hdaa	WHOLE DISK	no	7	0	173039

```
172031 1K blocks for divisions, 1008 1K blocks reserved for the system
```

divvy also displays information about block allocation for system tables and bad tracks.

You can change the name of the device with the 'n' command. *divvy* prompts you for the division number (from the *divvy* table displayed above), then for a new name.

The 'c' command causes a given division to become a new, empty filesystem when you exit from *divvy*. After using the 'c' command, you will see a 'yes' in the 'New File System?' column. If you use command 'p,' the 'yes' in the 'New File System?' column will change to a 'no', and the contents of the division will not change. The 'c' command must be used when changing the size of a filesystem.

With the 's' or 'start' command, you can start a division on a different block number. With the 'e' or 'end' command, you can end a division on a different block number.

You can use these commands to change the size of a partition. For example, if your disk is similar to the one in the sample *divvy* table above, and you want to make the **u** filesystem larger and the **swap** area smaller, do this:

> Make the swap area smaller with the 'e' command.
>
> Use the 's' command to make the **u** division bigger.
>
> Use the 'c' command to recreate the **u** filesystem.

Note that if any of the divisions overlap, *divvy* will complain when you try to exit and put you back in the menus to correct the situation.

The 'r' or 'restore' command restores the original partition table. This is useful if you make a serious mistake and want to return to where you started.

When you exit from *divvy*, you are prompted whether you want to save any changes you made, or exit without saving the changes. At this time, you can also go back to the *divvy* menu, and may also have the option to reinstall the original, default partition table. If you elect to save your changes, the new partition table will be written to the hard disk and any new filesystems (designated with the 'c' command) will be created.

See Also

badtrk(ADM), fdisk(ADM), fsck(ADM), fsname(ADM), hd(M), mkdev(C), mkfs(C), mknod(C)

Notes

divvy requires kernel level support from the device driver. If *divvy* lists the size of a disk as "0" blocks, or displays the following error messages, the device may not support dividing:

```
cannot read division table
```

or:

```
cannot get drive parameters
```

These errors may also occur if the prerequisite programs *dparam*, *fdisk* and *badtrk* are not run correctly.

If you change the size of filesystems (such as **/u**) after you have installed an AFS filesystem, you will have to use the 'c' command to re-create the filesystem and reinstall the files that are kept there. This is because the free list for that filesystem has changed. Be sure to backup the files in any filesystem you intend to change, using *backup*(ADM), *tar*(C), or *cpio*(C), before you run *divvy*. To change the size of the **root** filesystem, the operating system must be reinstalled.

During installation, if the filesystem on division 0 (generally root) becomes or remains large enough to require a scratch area during *fsck*, and one does not already exist, *divvy* prompts for whether one should be created. (The resulting filesystem, **/dev/scratch**, is used by *autoboot* if it runs *fsck*. **/dev/scratch** should also be entered when *fsck* prompts for a scratch file name, provided that the filesystem being checked is not larger than the root filesystem.) If all disk divisions have been used up, *divvy* will not prompt for a scratch filesystem, even if the root filesystem is large enough to require one.

This utility uses 512-byte blocks.

SCSI Conversion

The following table shows how the SCSI index number for a SCSI hard disk translates into its major number and minor number.

Device numbers are allocated to hard disks on a first-come-first-served basis, regardless of host adapter type, host adapter number controller number, or LUN.

Logical SCSI Index	Major Device Number	Base Minor Device Number
0	64	0
1	64	64
2	64	128
3	64	192
4	65	0
5	65	64
6	65	128
7	65	192
8	66	0
9	66	64
10	66	128
11	66	192
12	67	1
13	67	64
14	67	128
15	67	192
16	68	2
17	68	64
18	68	128
19	68	192
20	69	2
21	69	64
22	69	128
23	69	192
*	*	*
*	*	*
*	*	*
*	*	*
48	76	0
49	76	64
50	76	128
51	76	192

Value Added

divvy is an extension of AT&T System V provided in Altos UNIX System V.

dlayout

display hard disk partition, division, and size information

Syntax

/altos/bin/dlayout **[-hpds]** [*part_num*] [*device*]

Description

The *dlayout* utility displays the following hard disk configfuration information:

Partition
: (as configured with *fdisk*(ADM)) Displays parition number, active ststus, operating system type, start offset (in 512-byte blocks from the beginning of the hard disk), partition size (also in 512-byte blocks), and reserved area size.

Division
: (as configured with *divvy*(ADM)) Displays partition number, division number, start offset (in 1024-byte blocks *after* the reserved area, whihch is located at the beginning of the partition), and division size (in 1024-byte blocks).

Disk Size
: Displays raw disk capacity (in megabytes), number of cylinders, number of heads, and number of sectors per track.

The *part_num* (partition number) parameter can be used to restrict the display of information to the specified partition only. A valid partition number is any single digit between 1 and 4, or 5 for the current active partition.

The *device* paramter is used to specify which hard disk to display. The device should be a "whole-disk" physical device name (i.e., in the form **/dev/rhd**[*a-z*,*A-Z*]**0**). Do not use a division-level physical device name, such as **/dev/rhda10**. If *device* is not specified, then **/dev/rhda0** is assumed.

Options

-h Suppress title headers.

-p Display partition information only.

-d Display division information only.

-s Display disk size information only.

Note

On a SCSI hard disk, the displayed number of cylinders, heads, and sectors per track are computed using a special algorithm, and does not always reflect the actual hardware configuration.

See Also

divvy(ADM), fdisk(ADM), hd(HW), mkdev(ADM)

Value Added

dlayout(ADM) is an extension of AT&T System V provided in Altos UNIX System V.

dlvr_audit

produce audit records for subsystem events

Syntax

dlvr_audit [**-v**] tstamp event record pid cmd code [args ...]

Description

dlvr_audit is used by programs implementing protected subsystems as the means for sending audit records to the audit subsystem. Because those programs do not have the **writeaudit** privilege, they invoke *dlvr_audit* which sends the data over a message queue to the audit daemon, which appends the record to the audit trail. Because *dlvr_audit* is run as a child process of the process producing the record, it does not have the ability to write the audit device either. The message queue that it uses is only usable by the **audit** user, so *dlvr_audit* must be run SUID to the **audit** user. The group is inherited from the invoking process and is checked against those groups associated with protected subsystems. If the group cannot be identified with a protected subsystem, the record is ignored (so that general user programs cannot flood the audit subsystem with invalid messages).

The **-v** flag forces the program to report all of its actions. Normally, this flag is not used so that audit records can be made without the knowledge of the program user.

The required arguments apply to all audit records. The *tstamp* argument is the (ASCII number representation of the) time in seconds past Jan 1, 1970 that the audit record was produced. The *event* argument is the number of the event type as described in **<sys/audit.h>**. Similarly, the *record* argument is the audit record format type as described in **<sys/audit.h>**. The *pid* is the process ID of the event process. *Cmd* is the name of the protected subsystem command. *Code* is specific to the *event* type being generated.

There may be 0 or more optional arguments depending on the code. *dlvr_audit* uses the extra arguments to fill in specific fields required by the particular record format.

See Also

authaudit(S), audit(HW), ''Maintaining System Security,'' chapter of the *System Administrator's Guide*

Value Added

dlvr_audit is an extension of AT&T System V provided in Altos UNIX System V.

dmesg

displays the system messages on the console

Syntax

dmesg [-]

Description

The *dmesg* command displays all the system messages that have been generated since the last time the system was booted. If the option — is specified, it displays only those messages that have been generated since the last time the *dmesg* command was performed.

dmesg can be invoked periodically by placing instructions in the file **/usr/lib/crontab** . It can also be invoked automatically by the **/etc/rc2** scripts whenever the system is booted. See "Notes", below.

dmesg logs all error messages it prints in **/usr/adm/messages.** If *dmesg* is invoked automatically, the **messages** file continues to grow and can become very large. The system administrator should occasionally erase its contents.

Files

/etc/dmesg
/usr/adm/messages
/usr/adm/msgbuf

Notes

dmesg is included in this release for backwards compatibility only. The device **/dev/error** provides a more flexible means of logging error messages, and is recommended over *dmesg*. See *error*(M) for more information.

See Also

cron(C), error(M), messages(M)

dmesg was developed at the University of California, Berkeley, and is used with permission.

Value Added

dmesg is an extension of AT&T System V provided in Altos UNIX System V.

dparam

displays/changes hard disk characteristics

Syntax

dparam [**-w**]
dparam /dev/rhd[0| 1]0 [characteristics]

Description

The *dparam* command displays or changes the hard disk characteristics currently in effect. These changes go into effect immediately and are also written to the master boot block for subsequent boots. If a non-standard hard disk is used, this utility must be called before accessing the drive.

-w Causes a copy of **/etc/masterboot** to be copied to disk to ensure that non-standard hard disks are supported for the specified drive. This call must precede a call to write non-standard disk parameters for the desired parameters to be saved correctly in the masterboot block.

When called without options or disk characteristics, *dparam* prints the current disk characteristics (on the standard output) for the specified hard disk. These values are printed in the same order as the argument list.

When writing characteristics for the specified hard disk, *dparam* changes the current disk controller status and updates the masterboot block. The argument ordering is critical and must be entered as specified below. All characteristics must be entered when writing disk characteristics, otherwise an error is returned. Hard disk characteristics (in respective order) are:

number of cylinders
total number of cylinders on the hard disk

number of heads
number of heads

write cylinder
hardware specific, consult your hardware manual

write precompensation cylinder
hardware specific, consult your hardware manual

ecc number of bits of error correction on I/O transfers, consult your hardware manual

control — very hardware specific, consult your hardware manual

landing zone cylinder
: where to park heads after shutting down the system

number of sectors per track
: number of sectors per track on the hard disk

Examples

dparam -w

dparam /dev/rhd10

dparam /dev/rhd00 700 4 256 180 5 0 640 17

Notes

This utility changes the kernel's view of the hard disk parameters. It may be subject to restrictions imposed by the hardware configuration.

Value Added

dparam is an extension of AT&T System V provided in Altos UNIX System V.

fdisk

maintain disk partitions

Syntax

fdisk [[**-p**] [**-ad** partition] [**-c** partition start size] [**-f** devicename]]

Description

fdisk displays information about disk partitions. *fdisk* also creates and deletes disk partitions and changes the active partition. *fdisk* functionality is a superset of the MS-DOS command of the same name. *fdisk* is usually used interactively from a menu.

The hard disk has at most four partitions. Only one partition is active at any given time. It is possible to assign a different operating system to each partition. Once a partition is made active, the operating system resident in that partition boots automatically once the current operating system is halted.

The *fdisk* utility reports disk sizes in tracks. The number of tracks available on a hard disk is equal to the number of heads times the number of cylinders. The *fdisk* utility does not allocate the first track or the last cylinder on the hard disk when the ''Use Entire Disk for UNIX'' option is used. The first track on the hard disk is reserved for masterboot and the last cylinder is generally used when running hard disk diagnostics. You should not allocate the last cylinder if you plan to run diagnostics on your hard disk.

For example, if a disk has 4 heads and 615 cylinders, it has 2460 tracks, which *fdisk* reports as tracks 0-2459. If you choose the ''Use Entire Disk for UNIX'' option, *fdisk* will create a UNIX partition on tracks 1-2455. Track 0 is reserved for masterboot, and the last cylinder (tracks 2455-2459) is not assigned with the ''Use Entire Disk for UNIX'' option.

Partitions are defined by a ''partition table'' at the end of the master boot block. The partition table provides the location and size of the partitions on the disk. The partition table also defines the active partition. Each partition can be assigned to UNIX, DOS, or some other operating system. Once a DOS partition is set up, DOS files and directories resident in the DOS partition may be accessed while from the UNIX partition by means of the *dos*(C) commands. DOS may be booted without the DOS partition being active by entering ''**dos**'' at the boot prompt. See *boot*(HW).

Arguments

-p, -a, -d, -c
These flags are used to invoke *fdisk* non-interactively. The argument *number*, below, refers to a valid partition number (1-4).

-p Prints out the disk partition table, one partition to a line. For each partition, *fdisk* displays the following information: *partition start stop size status type.*

-a *number*
Activates partition *number.*

-d *number*
Deletes partition *number.*

-c *number start size*
Creates a partition, *number, size* tracks long beginning at track *start.* The **-c** option is used to use the entire disk for UNIX; the appending of a dash to the end of the command line accomplishes this, as in the following example:

```
fdisk -c 1 1 -
```

This syntax is used only during installation. If there are any existing partitions on the disk, this command will fail.

-f *name*
Open device *name* and read the partition table associated with that device's partition. The default is **/dev/rhda0.**

Options

When invoked interactively (without the **-p**, **-a**, **-d**, or **-c** options), *fdisk* displays a prompt and a menu of five options.

1. Display Partition Table.
   This option displays a table of information about each partition on the hard disk. The PARTITION column gives the partition number. The STATUS column tells whether the partition is active (A) or inactive (I). TYPE tells whether the partition is a UNIX partition, a DOS partition, or "other". The option also displays the starting track, ending track and total number of tracks in each partition.

2. Use Entire Disk for UNIX.
   fdisk creates one partition that includes all the tracks on the disk, except the first track and the last cylinder. This partition is assigned to the UNIX system and is designated the active partition.

3. Use Rest of Disk for UNIX.
   fdisk creates one partition that occupies the remainder of the disk. This partition is assigned to UNIX and is designated the active partition.

4. Create UNIX Partition
   This option allows the creation of a partition by altering the partition table. *fdisk* reports the number of tracks available for each partition and the number of tracks in use. *fdisk* prompts for the partition to create, the starting track and size in tracks.

5. Activate Partition
   This option activates the specified partition. Only one partition may be active at a time. The operating system residing in the newly activated partition boots once the current operating system is halted.

6. Delete Partition
   This option requests which partition you wish to delete. *fdisk* reports the new available amount of disk space in tracks.

Exit the *fdisk* program by typing a 'q' at the main *fdisk* menu.

Notes

The minimum recommended size for a UNIX partition is 50 megabytes.

Since *fdisk* is intended for use with DOS, it may not work with all operating system combinations.

See Also

dos(C), hd(HW)

Value Added

fdisk is an extension of AT&T System V provided in Altos UNIX System V.

fdswap

swaps default boot floppy drive

Syntax

fdswap [on|off]

Description

fdswap tells the CMOS to swap the default floppy drive used to read boot information at boot time. For example, if your computer defaults to read boot information on drive A, *fdswap on* changes the default drive to drive B.

fdswap with no arguments reports the current *fdswap* state, on or off. *fdswap off* switches the drive setting back to the default configuration. Changing the drives take effect on the next boot of the system.

Notes

Support for this functionality is only available on a small number of machines. The ROMs must recognize and interpret the CMOS flag that specifies that the floppy drives are swapped.

ff

list file names and statistics for a filesystem

Syntax

/etc/ff [options] special

Description

The *ff* command reads the i-list and directories of the *special* file, assuming it is a file system. Inode data is saved for files which match the selection criteria. Output consists of the path name for each saved inode, plus other file information requested using the print *options* below. Output fields are positional. The output is produced in inode order; fields are separated by tabs. The default line produced by *ff* is:

path-name i-number

With all *options* enabled, output fields would be:

path-name i-number size uid

The argument *n* in the *option* descriptions that follow is used as a decimal integer (optionally signed), where +*n* means more than *n*, -*n* means less than *n*, and *n* means exactly *n*. A day is defined as a 24-hour period.

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

-l Generate a supplementary list of all path names for multiple-linked files.

-p *prefix* The specified *prefix* will be added to each generated path name. The default is . (dot).

-s Print the file size, in bytes, after each path name.

-u Print the owner's login name after each path name.

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

-m *n* Select if the inode has been modified in *n* days.

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

-n *file* Select if the inode has been modified more recently than the argument *file*.

-i *inode-list* Generate names for only those inodes specified in *inode-list*.

See Also

find(C), ncheck(ADM)

Notes

If the **-l** option is not specified, only a single path name out of all possible ones is generated for a multiple-linked inode. If **-l** is specified, all possible names for every linked file on the file system are included in the output. However, no selection criteria apply to the names generated.

This command only works on UNIX filesystems.

fixperm

correct or initialize file permissions and ownership

Syntax

fixperm [**-cfgilnpsvwDS** [**-d** package]] specfile

Description

For each line in the specification file **specfile,** *fixperm* makes the listed pathname conform to a specification. *fixperm* is typically used to configure a UNIX system upon installation. It can only be invoked by a superuser, and it only works from the root directory. If it is invoked from any other directory, incorrect results will be returned.

The specification file has the following format: Each non-blank line consists of either a comment or an item specification. A comment is any text from a pound sign "#" up to the end of the line. There is one item specification per line. User and group id numbers must be specified at the top of the specification file for each user and group mentioned in the file. The syntax for the definition section is simple: the first field indicates the type of id (either *uid* or *gid*), the second contains the name reference for the id, and the third is the corresponding numeric id. Example:

```
uid     root    0
```

An item specification consists of a package specifier, a permission specification, owner and group specifications, the number of links on the file, the file name, and an optional volume number.

The package specifier is an arbitrary string which is the name of a package within a distribution set. A package is a set of files.

After the package specifier is a permission specification. The permission specification consists of a file type, followed by a numeric permission specification. The item specification is one of the following characters:

x Executable.

a Archive.

e Empty file (create if **-c** option given).

b Block device.

c Character device.

d Directory.

f Text file.

p Named pipe.

o OK. It indicates to *fixperm* that there should be no file type checking allowing any format or contents in what would normally be the header section of an executable. For example, data files and encrypted files should be of type "o".

If the item specification is used as an upper-case letter, then the file associated with it is optional, and *fixperm* will not return an error message if it does not exist.

The numeric permission conforms to the scheme described in *chmod*(C). The owner and group permissions are in the third column separated by a slash: e.g.,: "bin/bin". The fourth column indicates the number of links. If there are links to the file, the next line contains the linked filename with no other information. The fifth column is a pathname. The pathname must be relative, i.e., not preceded by a slash "/". The sixth column is only used for special files, giving the major and minor device numbers, or volume numbers.

Options

The following options are available from the command line, unless otherwise noted:

-c Create empty files and missing directories. Also or creates (or modifies) device files.

-g Instructs *fixperm* to list devices as specified in the permlist (similar to the **-f** flag, which lists files on standard output). No changes are made as a result of this flag.

-d *package*
Process input lines beginning with given package specifier string (see above). For instance, **-d**BASE processes only items specified as belonging to the Basic utilities set. The default action is to process all lines.

-u *package*
Like **-u**, but processes items that are not part of the given package.

-f List files only on standard output. Does not modify target files.

-i (Available from a program or shell script only)

Check only if the selected packages are installed. Return values are:

0:	package completely installed
4:	package not installed
5:	package partially installed

-l List files and directories on standard output. Does not modify target files.

-n Report errors only. Does not modify target files.

-p Override default uid/gid found in **/etc/passwd** and **/etc/group** with the value found in the permlist. Because UNIX and XENIX have different values for certain uid and gids (for example, in UNIX bin=2, and XENIX bin=3) the default value is gleaned from the **/etc/passwd** and **/etc/group** files. This option forces the values to be taken from the perms list. It also generates a warning if the permlist doesn't **/etc/passwd** and **/etc/group**.

-D List directories only on standard output. Does not modify target files.

-v Verbose, in particular, issues a complaint if executable files are word swapped, not fixed stack, not separate I and D, or not stripped.

-s Modify special device files in addition to the rest of the permlist.

-w Lists where (what volume) the specified files or directories are located.

-S Issues a complaint if files are not in x.out format.

The following two lines make a distribution and invoke *tar*(C) to archive only the files in perms.inst on /dev/sample:

```
/etc/fixperm -f /etc/perms/inst > list
tar cfF /dev/sample list
```

This example reports *BASE* package errors:

/etc/fixperm -nd BASE /etc/perms/*

or

/etc/fixperm -nd BASE /etc/perms/*filename*

Notes

Usually *fixperm* is only run by a shell script at installation.

See Also

custom (ADM)

Value Added

fixperm is an extension of AT&T System V provided in Altos UNIX System V.

fsave

interactive, error-checking filesystem backup

Synopsis

fsave filesystem [backupinfo] [mediainfo] [sitename]

Description

fsave is used by *fsphoto*(ADM) to provide a semi-automated interface to *xbackup*(ADM) and *cpio*(C) for backing-up filesystems. Human intervention is required to mount and dismount tapes or floppies at the appropriate times, but is kept to a minimum to reduce the potential for error.

The operator is prompted each time some action is required, such as mounting or unmounting a tape or floppy. These prompts, and their possible selections, are described below.

For all prompts, an answer of **h**, **H**, or **?** will display a short summary of the possible answers.

Filesystem dump (backup)

The following prompt displays the defaults (gleaned from the *schedule* database file) and presents options to alter them:

```
Level dumplevel dump of filesystem filesystem , date
        media size: size feet [or Kb]
        media drive: drive
This media will be saved for howlong, and is howvital.

M)ounted volume, P)ostpone, C)heck or F)ormat volumes, R) Retension or H)elp:
```

The values displayed dictate the following instructions: *filesystem* is to be backed-up using *size*-foot long magtapes (or *size*-kilobyte big floppies) mounted on drive *drive*. The *media* will be saved for *howlong* ("1 year," "2 months," etc.), and being a level *dumplevel* dump, is *howvital* ("critical," "precautionary," etc.).

The menu options are:

m A volume of the asked for *size* has been mounted (write-enabled), so begin the dump.

m*newsize* Insufficient volumes of the originally asked for size are available, so a *newsize* big volume has been mounted instead. If the dump extends across more than one volume, each volume must be of the same *size*.

p Postpone this backup until later (*fsphoto* will automatically retry this *filesystem* next time it is run).

c Recheck the volumes used to backup *filesystem* for errors. This answer is useful when a dump mysteriously fails and *fsave* is starting over from the beginning, but the operator doesn't believe there really is a problem (for example, the tape drive was accidentally left offline or the floppy door was left open), and wants to check the volumes again.

f Format the currently mounted volume (useful mainly for floppies).

r Retension cartridge tape using /usr/bin/tape.

If multiple volumes are required, *backup* will pause for the next volume to be mounted. Be certain to keep track of the volume order.

Format check

The format of ''critical'' volumes are checked using *dumpdir*(ADM):

```
Check vital volumes for format errors
M)ounted first volume, S)kip format check, or H)elp:
```

The menu options are:

m The first volume has been (or still is) mounted, and *dumpdir* can now check the volume format.

s Skip checking the volume format, and continue on to the read error check (below).

The format is not always checked, but when it is, the first volume written must be mounted.

Read error check

All volumes created using *xbackup*(ADM) are read using *xrestore*(ADM), which checks for errors during reading. If an error occurs, the dump is declared unsuccessful and is retried from the beginning.

```
Check vital volumes for read errors
M)ounted which volume, E)rror on previous volume, D)one, S)kip checks, or H)elp:
```

The menu options are:

m The *which* (''first'' or ''next'') volume has been mounted on the drive and is ready to be checked for read errors.

e An error occurred on the last volume checked, and the dump should be retried.

d All volumes have been checked and no errors occurred, so the filesystem has been successfully backed-up; This backup is done.

s Don't bother (skip) checking the rest of the volumes for read errors.

Every volume should be checked for read errors; *xrestore* requires the volumes to be checked in first-to-last order. Volumes that produce read errors should be marked "suspect," discarded and the dump run once again.

After the backup has been successfully performed, instructions are given on how to label the volumes.

Arguments

fsave is normally run by *fsphoto*, which passes all the proper arguments based on the *schedule* (ADM) database.

filesystem
The filesystem to be backed-up.

dumpinfo
A set of blank-separated strings that give some optional information about this backup:

dumplevel size savetime importance marker

Each of these component strings may be quoted and can thus contain spaces.

dumplevel The level of the dump to be performed. This is a single digit from **0** to **9** (passed to *dump*), or the letter **x** (which means no dump is to be done). The default is to perform a level 0 dump.

size The size of the media volumes that should be used. This should be in feet for tapes and kilobytes for floppies. A *size* of - means to use the first size listed in *mediainfo*. This is the default.

savetime How long this backup is to be saved (for example, "3 months"). Default is "1 year."

importance
How important is this backup? (For example, "critical" or "precautionary.") Those which are "critical" have their format checked by *dumpdir*. Default is

"important."

marker Either "none" (the default) or an additional label to place on each volume (for example, "a pink sticker").

A typical *dumpinfo* might look like:

```
9 1200 "2 weeks" useful "a blue X"
```

which specifies that a level 9 dump is to be done on a 1200 foot tape (or 1200 kilobyte floppy) which will be saved for 2 weeks and is to be marked with a blue cross (in addition to a more descriptive label). This backup is merely considered "useful" and thus will not be checked by *dumpdir*.

mediainfo
A set of blank-separated strings that give some optional information about this the media to be used:

drive **d** *density sizes* ... [*format*]
drive **k** *sizes* ... [*format*]

drive The name of backup device to use. The default is */dev/rmt0*.

k *sizes...* If **k** is specified, *drive* is assumed to be a floppy, and the list of *sizes* which follow define the allowable capacities of the floppies that can be used (in kilobytes).

d *density sizes...*
Otherwise, **d** must be specified. In this case, *drive* is assumed to be a magtape at *density* BPI , in one of the possible *sizes* (in feet).

format The UNIX command used to format the tape or floppy so described.

A *mediainfo* describing 9-track magtape would be:

```
media /dev/rmt0 d 1600 2400 1200 600
media /dev/rmt2 d 800  1400 1200 600
```

which specifies that */dev/rmt0* is a 1600 BPI magtape capable of handling 2400, 1200, and 600 foot reels, and that */dev/rmt2* is the 800 BPI device.

A floppy might be described with:

```
media /dev/fd0 k 1024 format /dev/fd0
```

which describes device */dev/fd0* as a megabyte (1024 kilobytes) floppy formatted by the command:

format /dev/fd0

sitename
Where this backup was made (for example, the name of the company or which building). Note that the *uucp*(C) nodename from */etc/systemid* is automatically placed on the volume labels.

Only the super-user can execute the *fsave* command.

Files

/etc/systemid
Name of this machine.

/etc/ddate
backup-maintained record of last time each filesystem was backed-up.

/dev/tty
Always-existent character-special device.

See Also

fsphoto(ADM), schedule(ADM), xbackup(ADM), dumpdir(ADM), xrestore(ADM), cpio(C), basename(C)

Diagnostics

A successful backup exits successfully (0), but errors generate a complaint and an exit status of 1. *fsave* complains about illegal or incorrect arguments, and exits with a status of 2.

If the backup of *filesystem* is postponed, *fsave* exits with a status of 3.

Value Added

fsave is an extension of AT&T System V provided in Altos UNIX System V.

fsck, dfsck

checks and repairs filesystems

Syntax

/etc/fsck [options] [filesystem] ...

/etc/dfsck options1] filesys1 ... -[options2] filesys2 ...

Description

fsck audits and interactively repairs inconsistent conditions for all supported filesystems. If the filesystem is consistent, the the number of files, number of blocks used, and number of blocks free are reported. If the filesystem is inconsistent, the operator is prompted for concurrence before each correction is attempted. It should be noted that most corrective actions result in some loss of data. The amount and severity of the loss may be determined from the diagnostic output. (An experienced operator can resolve discrepancies manually using *fsdb*(ADM), the filesystem debugger.) The default action for each consistency correction is to wait for the operator to respond "yes" or "no". If the operator does not have write permission *fsck* defaults to the action of the **-n** option.

The following flags are interpreted by *fsck*:

-C[*clustersize*]
: Converts the named S51K filesystem into an AFS (Acer Fast Filesystem). The **-s** option must also be present. The *clustersize* argument must be a power of 2 and less than 16 (8 is the recommended value). The increase in speed that is possible with a fast filesystem will not be immediately apparent; it will take affect only with the new files added to the filesystem. There is little or no benefit in transforming a filesystem that is nearly full; if it is within a few blocks of being full, the conversion will not work. (This option can only be used to convert an S51K fileystem.)

-y Assumes a yes response to all questions asked by *fsck*.

-n Assumes a no response to all questions asked by *fsck*; do not open the filesystem for writing.

-s*b*:*c* Ignores the actual free list and (unconditionally) reconstructs a new one by rewriting the super-block of the filesystem. The filesystem *must* be unmounted while this is done.

The -*sb*:*c* option allows for creating an optimal free-list organization. The following forms are supported:

-s
-sBlocks-per-cylinder:Blocks-to-skip (filesystem interleave)
(for anything else)

If *b*:*c* is not given, then the values used when the filesystem was created are used. If these values were not specified, then a reasonable default value is used.

-S Conditionally reconstructs the free list. This option is like -*sb*:*c* above except that the free list is rebuilt only if there are no discrepancies discovered in the filesystem. Using **-S** forces a "no" response to all questions asked by *fsck*. This option is useful for forcing free list reorganization on uncontaminated filesystems.

-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. Make certain you leave a space between the **-t** and the filename, or *fsck* will use the entire filesystem as a scratch file and erase the entire disk. If you created a scratch filesystem during installation then you can use /dev/scratch as the filename, provided that the filesystem being checked is no larger than the **root** filesystem. Without the **-t** flag, *fsck* prompts the operator for the name of the scratch file. The file chosen should not be on the filesystem being checked, and if it is not a special file or did not already exist, it is removed when *fsck* completes. If the system has a large hard disk there may not be enough space on another filesystem for the scratch file. In such cases, if the system has a floppy drive, use a blank, formatted floppy in the floppy drive with (for example) **/dev/fd0** specified as the scratch file.

-q Quiet *fsck*. Do not print size-check messages in Phase 1. Unreferenced **fifo5** files will selectively 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 (Phase 1) and check the free list (Phase 5). The free list will be reconstructed (Phase 6) if it is necessary.

-b Recovers a UNIX root filesystem. The required *filesystem* argument must refer to the root filesystem, and preferably to the block device (normally **/dev/root**). This switch implies **-y** and overrides **-n** . If any modifications to the filesystem are required, the filesystem will be automatically mounted.

-rr Recover XENIX root filesystem. Equivanlent to the **-b** option described above for UNIX root filesystem recovery.

-c Causes any supported filesystem to be converted to the type of the current filesystem. The user is prompted to verify the request for each filesystem that requires conversion unless the -y option is specified. It is recommended that every filesystem be checked with this option while unmounted. To update the active root filesystem, it should be checked with:

fsck -c -rr /dev/root

If no *filesystems* are specified, *fsck* reads a list of default filesystems from the file **/etc/checklist.**

Inconsistencies checked are as follows:

- Blocks claimed by more than one inode or the free list
- Blocks claimed by an inode or the free list outside the range of the filesystem
- Incorrect link counts
- Size checks:
  Incorrect number of blocks
  Directory size not 16-byte aligned
- Bad inode format
- Blocks not accounted for anywhere
- Directory checks:
  File pointing to unallocated inode
  Inode number out of range
- Super block checks:
  More than 65536 inodes
  More blocks for inodes than there are in the filesystem
- Bad free block list format
- Total free block or free inode count incorrect

Orphaned files and directories (allocated but unreferenced) are, with the operator's concurrence, reconnected by placing them in the **lost+found** directory. The name assigned is the inode number. The only restriction is that the directory **lost+found** must preexist in the root of the filesystem 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).

dfsck allows two filesystem checks on two different drives simultaneously. *Options1* and *options2* are used to pass options to *fsck* for the two sets of filesystems. A - is the separator between filesystem groups.

The *dfsck* program permits an operator to interact with two *fsck* programs at once. To help in this, *dfsck* displays the filesystem name for each message to the operator. When answering a question from *dfsck*, the operator must preface the response with a **1** or a **2** (indicating that the answer refers to the first or second filesystem group).

Do not use *dfsck* to check the *root* filesystem.

Files

/etc/checklist	Contains default list of filesystems to check
/etc/default/boot	Automatic boot control

See Also

autoboot(ADM), fsdb(ADM), checklist(F), filesystem(F), init(M)

Notes

The directory **/etc/fscmd.d/***TYPE* contains programs for each file system type; each of these programs applies some appropriate heuristic to determine whether the supplied *special* file is of the type for which it checks.

fsck will not run on a mounted non-raw filesystem unless the filesystem is the root filesystem or unless the **-n** option is specified and no writing out of the filesystem will take place. If any such attempt is made, a warning is displayed and no further processing of the filesystem is done for the specified device.

Although checking a raw device is almost always faster, there is no way to tell if the filesystem is mounted. And cleaning a mounted filesystem will almost certainly result in an inconsistent superblock.

Warning

Filesystems created under UNIX-86 version 3.0 are not supported under UNIX System V/386 3.2 because the word ordering in type *long* variables has changed. *fsck* is capable of auditing and repairing UNIX version 3.0 file systems if the word ordering is correct.

For the root filesystem, "fsck -rr /dev/root" should be run. For all other filesystems, "fsck /dev/??" on the *unmounted* block device should be used.

Diagnostics

Initialization Phase

Command syntax is checked. Before the filesystem check can be performed, **fsck** sets up certain tables and opens some files. The **fsck** terminates on initialization errors.

General Errors

Three error messages may appear in any phase. While they seem to offer the option to continue, it is generally best to regard them as fatal, end the run, and investigate what may have caused the problem.

CAN NOT SEEK: BLK B (CONTINUE?)
: The request to move to a specified block number *B* in the filesystem failed. The occurrence of this error condition indicates a serious problem (probably a hardware failure) that may require additional help.

CAN NOT READ: BLK B (CONTINUE?)
: The request for reading a specified block number *B* in the filesystem failed. The occurrence of this error condition indicates a serious problem (probably a hardware failure) that may require additional help.

CAN NOT WRITE: BLK B (CONTINUE?)
: The request for writing a specified block number *B* in the filesystem failed. The disk may be write-protected.

Meaning of Yes/No Responses

Prompt	n(no)	y(yes)
CONTINUE?	Terminates program. (This is the recommended response.)	Attempts to continue to run filesystem check. Often, however, the problem persists. The error condition does not allow a complete check of the filesystem. A second run of *fsck* should be made to recheck this filesystem.

Phase 1: Check Blocks and Sizes

This phase checks the inode list.

Meaning of Yes/No Responses—Phase 1

Prompt	n(no)	y(yes)
CONTINUE?	Terminates the program. (Recommended response.)	Continues with the program. This error condition means that a complete check of the filesystem is not possible. A second run of *fsck* should be made to recheck this filesystem.
CLEAR?	Ignores the error condition. A NO response is only appropriate if the user intends to take other measures to fix the problem.	Deallocates i-node *I* by zeroing its contents. This may invoke the UNALLOCATED error condition in Phase 2 for each directory entry pointing to this i-node.

Phase 1 Error Messages

UNKNOWN FILE TYPE I=I (CLEAR?)
: The mode word of the i-node *I* suggests that the i-node is not a pipe, special character i-node, regular i-node, or directory i-node.

LINK COUNT TABLE OVERFLOW (CONTINUE?)
: An internal table for *fsck* containing allocated i-nodes with a link count of zero has no more room.

B BAD I=I
: I-node *I* contains block number *B* with a number lower than the number of the first data block in the filesystem or greater than the number of the last block in the filesystem. This error condition may invoke the EXCESSIVE BAD BLKS error condition in Phase 1 if i-node *I* has too many block numbers outside the filesystem range. This error condition invokes the BAD/DUP error condition in Phase 2 and Phase 4.

EXCESSIVE BAD BLOCKS I=I (CONTINUE?)
: There is more than a tolerable number (usually 10) of blocks

with a number lower than the number of the first data block in the filesystem or greater than the number of the last block in the filesystem associated with i-node *I*.

B DUP I=I
: I-node *I* contains block number *B*, which is already claimed by another i-node. This error condition may invoke the EXCESSIVE DUP BLKS error condition in Phase 1 if i-node *I* has too many block numbers claimed by other i-nodes. This error condition invokes Phase 1B and the BAD/DUP error condition in Phase 2 and Phase 4.

EXCESSIVE DUP BLKS I=I (CONTINUE?)
: There is more than a tolerable number (usually 10) of blocks claimed by other i-nodes.

DUP TABLE OVERFLOW (CONTINUE?)
: An internal table in *fsck* containing duplicate block numbers has no more room.

POSSIBLE FILE SIZE ERROR I=I
: The i-node *I* size does not match the actual number of blocks used by the i-node. This is only a warning. If the *-q* option is used, this message is not printed.

DIRECTORY MISALIGNED I=I
: The size of a directory i-node is not a multiple of 16. This is only a warning. If the *-q* option is used, this message is not printed.

PARTIALLY ALLOCATED INODE I=I (CLEAR?)
: I-node *I* is neither allocated nor unallocated.

Phase 1B: Rescan for More DUPS

When a duplicate block is found in the filesystem, the filesystem is rescanned to find the i-node that previously claimed that block. When the duplicate block is found, the following information message is printed:

B DUP I=I
: I-node *I* contains block number *B*, which is already claimed by another i-node. This error condition invokes the BAD/DUP error condition in Phase 2. I-nodes with overlapping blocks may be determined by examining this error condition and the DUP error condition in Phase 1.

Phase 2: Check Path Names

This phase removes directory entires pointing to bad inodes found in Phase 1 and phase 1B.

Meaning of Yes/No Responses—Phase 2

Prompt	n(no)	y(yes)
FIX?	Terminates the program since *fsck* will be unable to continue.	In Phase 2, a y(yes) response to the FIX? prompt says: Change the root i-node type to "directory." If the root i-node data blocks are not directory blocks, a very large number of error conditions are produced.
CONTINUE?	Terminates the program.	Ignores DUPS/BAD error condition in root i-node and attempt to continue to run the filesystem check. If root i-node is not correct, then this may result in a large number of other error conditions.
REMOVE?	Ignores the error condition. A NO response is only appropriate if the user intends to take other measures to fix the problem.	Removes duplicate or unallocated blocks.

Phase 2 Error Messages

ROOT INODE UNALLOCATED. TERMINATING

The root i-node (always i-node number 2) has no allocate mode bits. The occurrence of this error condition indicates a serious problem. The program stops.

ROOT INODE NOT DIRECTORY (FIX?)
The root i-node (usually i-node number 2) is not directory i-node type.

DUPS/BAD IN ROOT INODE (CONTINUE?)
Phase 1 or Phase 1B found duplicate blocks or bad blocks in the root i-node (usually i-node number 2) for the filesystem.

I OUT OF RANGE I=I NAME=F (REMOVE?)
A directory entry *F* has an i-node number *I* that is greater than the end of the i-node list.

UNALLOCATED I=I OWNER=O MODE=M SIZE=S MTIME=T NAME=F (REMOVE?)
A directory entry *F* has an i-node *I* without allocate mode bits. The owner *O*, mode *M*, size *S*, modify time *T*, and filename *F* are printed. If the filesystem is not mounted and the *-n* option was not specified, the entry is removed automatically if the i-node it points to is character size 0.

DUP/BAD I=I OWNER=O MODE=M SIZE=S MTIME=T DIR=F (REMOVE?)
Phase 1 or Phase 1B found duplicate blocks or bad blocks associated with directory entry *F*, directory i-node *I*. The owner *O*, mode *M*, size *S*, modify time *T*, and directory name *F* are printed.

DUP/BAD I=I OWNER=O MODE=M SIZE=S MTIME=T FILE=F (REMOVE?)
Phase 1 or Phase 1B found duplicate blocks or bad blocks associated with file entry *F*, i-node *I*. The owner *O*, mode *M*, size *S*, modify time *T*, and filename *F* are printed.

BAD BLK B IN DIR I=I OWNER=O MODE=M SIZE=S MTIME=T
This message only occurs when the *-D* option is used. A bad block was found in DIR i-node *I*. Error conditions looked for in directory blocks are nonzero padded entries, inconsistent "." and ".." entries, and embedded slashes in the name field. This error message means that the user should at a later time either remove the directory i-node if the entire block looks bad or change (or remove) those directory entries that look bad.

Phase 3: Check Connectivity

This phase is concerned with the directory connectivity seen in Phase 2.

Meaning of Yes/No Responses—Phase 3

Prompt	n(no)	y(yes)
RECONNECT?	Ignores the error condition. This invokes the UNREF error condition in Phase 4. A NO response is only appropriate if the user intends to take other measures to fix the problem.	Reconnects directory i-node *I* to the filesystem in directory for lost files (usually *lost+found*). This may invoke a *lost+found* error condition if there are problems connecting directory i-node *I* to *lost+found*. This invokes CONNECTED information message if link was successful.

Phase 3 Error Messages

UNREF DIR I=I OWNER=O MODE=M SIZE=S MTIME=T (RECONNECT?)

The directory i-node *I* was not connected to a directory entry when the filesystem was traversed. The owner *O*, mode *M*, size *S*, and modify time *T* of directory i-node *I* are printed. The *fsck* program forces the reconnection of a nonempty directory.

SORRY. NO lost+found DIRECTORY

There is no *lost+found* directory in the root directory of the filesystem; *fsck* ignores the request to link a directory in *lost+found*. This invokes the UNREF error condition in Phase 4. Possible problem with access modes of *lost+found*.

SORRY. NO SPACE IN lost+found DIRECTORY

There is no space to add another entry to the *lost+found* directory in the root directory of the filesystem; *fsck* ignores the request to link a directory in *lost+found*. This invokes the UNREF error condition in Phase 4. Clean out unnecessary entries in *lost+found* or make *lost+found* larger (see Procedure 5.2).

DIR I=I1 CONNECTED. PARENT WAS I=I2

This is an advisory message indicating a directory i-node *I1* was successfully connected to the *lost+found* directory. The parent i-node *I2* of the directory i-node *I1* is replaced by the i-node number of the *lost+found* directory.

Phase 4: Check Reference Counts

This phase checks the link count information seen in Phases 2 and 3.

Meaning of Yes/No Responses—Phase 4

Prompt	n(no)	y(yes)
RECONNECT?	Ignores this error condition. This invokes a CLEAR error condition later in Phase 4.	Reconnect i-node *I* to filesystem in the directory for lost files (usually *lost+found*). This can cause a *lost+found* error condition in this phase if there are problems connecting i-node *I* to *lost+found.*
CLEAR?	Ignores the error condition. A NO response is only appropriate if the user intends to take other measures to fix the problem.	Deallocates the i-node by zeroing its contents.
ADJUST?	Ignores the error condition. A NO response is only appropriate if the user intends to take other measures to fix the problem.	Replaces link count of file i-node *I* with *Y*.
FIX?	Ignores the error condition. A NO response is only appropriate if the user intends to take other measures to fix the problem.	Replaces count in super-block by actual count.

Phase 4 Error Messages

UNREF FILE I=I OWNER=O MODE=M SIZE=S MTIME=T (RECONNECT?)

I-node *I* was not connected to a directory entry when the filesystem was traversed. The owner *O*, mode *M*, size *S*, and modify time *T* of i-node *I* are printed. If the *-n* option is omitted and the filesystem is not mounted, empty files are cleared automatically. Nonempty files are not cleared.

SORRY. NO lost+found DIRECTORY

There is no *lost+found* directory in the root directory of the filesystem; *fsck* ignores the request to link a file in *lost+found.* This invokes the CLEAR error condition later in Phase 4. Possible problem with access modes of *lost+found.*

SORRY. NO SPACE IN lost+found DIRECTORY

There is no space to add another entry to the *lost+found* directory in the root directory of the filesystem; *fsck* ignores the request to link a file in *lost+found.* This invokes the CLEAR error condition later in Phase 4. Check size and contents of *lost+found.*

(CLEAR)

The i-node mentioned in the immediately previous UNREF error condition cannot be reconnected.

LINK COUNT FILE I=I OWNER=O MODE=M SIZE=S MTIME=T COUNT=X SHOULD BE Y (ADJUST?)

The link count for i-node *I*, which is a file, is *X* but should be *Y*. The owner *O*, mode *M*, size *S*, and modify time *T* are printed.

LINK COUNT DIR I=I OWNER=O MODE=M SIZE=S MTIME=T COUNT=X SHOULD BE Y (ADJUST?)

The link count for i-node *I*, which is a directory, is *X* but should be *Y*. The owner *O*, mode *M*, size *S*, and modify time *T* of directory i-node *I* are printed.

LINK COUNT F I=I OWNER=O MODE=M SIZE=S MTIME=T COUNT=X SHOULD BE Y (ADJUST?)

The link count for *F* i-node *I* is *X* but should be *Y*. The filename *F*, owner *O*, mode *M*, size *S*, and modify time *T* are printed.

UNREF FILE I=I OWNER=O MODE=M SIZE=S MTIME=T (CLEAR?)

I-node *I*, which is a file, was not connected to a directory entry when the filesystem was traversed. The owner *O*, mode *M*, size *S*, and modify time *T* of i-node *I* are printed. If the *-n* option is omitted and the filesystem is not mounted, empty files are cleared automatically. Nonempty directories are not cleared.

UNREF DIR I=I OWNER=O MODE=M SIZE=S MTIME=T (CLEAR?)

I-node *I*, which is a directory, was not connected to a directory entry when the filesystem was traversed. The owner *O*, mode *M*, size *S*, and modify time *T* of i-node *I* are printed. If the *-n* option is omitted and the filesystem is not mounted, empty directories are cleared automatically. Nonempty directories are not cleared.

BAD/DUP FILE I=I OWNER=O MODE=M SIZE=S MTIME=T (CLEAR?)

Phase 1 or Phase 1B found duplicate blocks or bad blocks associated with file i-node *I*. The owner *O*, mode *M*, size *S*, and modify time *T* of i-node *I* are printed.

BAD/DUP DIR I=I OWNER=O MODE=M SIZE=S MTIME=T (CLEAR?)

Phase 1 or Phase 1B found duplicate blocks or bad blocks associated with directory i-node *I*. The owner *O*, mode *M*, size *S*, and modify time *T* of i-node *I* are printed.

FREE INODE COUNT WRONG IN SUPERBLK (FIX?)

The actual count of the free i-nodes does not match the count in the super-block of the filesystem. If the *-q* option is specified, the count will be fixed automatically in the super-block.

Phase 5: Check Free List

This phase checks the free-block list.

Meaning of Yes/No Responses—Phase 5

Prompt	n(no)	y(yes)
CONTINUE?	Terminates the program.	Ignores rest of the free-block list and continue execution of *fsck*. This error condition will always invoke BAD BLKS IN FREE LIST error condition later in Phase 5.

(Continued)

Prompt	n(no)	y(yes)
FIX?	Ignores the error condition. A NO response is only appropriate if the user intends to take other measures to fix the problem.	Replaces count in super-block by actual count.
SALVAGE?	Ignores the error condition. A NO response is only appropriate if the user intends to take other measures to fix the problem.	Replaces actual free-block list with a new free-block list. The new free-block list will be ordered according to the gap and cylinder specs of the *-s* or *-S* option to reduce time spent waiting for the disk to rotate into position.

Phase 5 Error Messages

EXCESSIVE BAD BLKS IN FREE LIST (CONTINUE?)
The free-block list contains more than a tolerable number (usually 10) of blocks with a value less than the first data block in the filesystem or greater than the last block in the filesystem.

EXCESSIVE DUP BLKS IN FREE LIST (CONTINUE?)
The free-block list contains more than a tolerable number (usually 10) of blocks claimed by i-nodes or earlier parts of the free-block list.

BAD FREEBLK COUNT
The count of free blocks in a free-list block is greater than 50 or less than 0. This error condition will always invoke the BAD FREE LIST condition later in Phase 5.

X BAD BLKS IN FREE LIST
X blocks in the free-block list have a block number lower than the first data block in the filesystem or greater than the last block in the filesystem. This error condition will always invoke the BAD FREE LIST condition later in Phase 5.

X DUP BLKS IN FREE LIST

X blocks claimed by i-nodes or earlier parts of the free-list block were found in the free-block list. This error condition will always invoke the BAD FREE LIST condition later in Phase 5.

X BLK(S) MISSING

X blocks unused by the filesystem were not found in the free-block list. This error condition will always invoke the BAD FREE LIST condition later in Phase 5.

FREE BLK COUNT WRONG IN SUPERBLOCK (FIX?)

The actual count of free blocks does not match the count in the super-block of the filesystem.

BAD FREE LIST (SALVAGE?)

This message is always preceded by one or more of the Phase 5 information messages. If the *-q* option is specified, the free-block list will be salvaged automatically.

Phase 6: Salvage Free List

This phase reconstructs the free-block list. It has one possible error condition that results from bad blocks-per-cylinder and gap values.

Phase 6 Error Messages

DEFAULT FREE-BLOCK LIST SPACING ASSUMED

This is an advisory message indicating the blocks-to-skip (gap) is greater than the blocks-per-cylinder, the blocks-to-skip is less than 1, the blocks-per-cylinder is less than 1, or the blocks-per-cylinder is greater than 500. The values of 7 blocks-to-skip and 400 blocks-per-cylinder are used.

Cleanup Phase

Once a filesystem has been checked, a few cleanup functions are performed. The cleanup phase displays advisory messages about the filesystem and status of the filesystem.

Cleanup Phase Messages

X files Y blocks Z free

This is an advisory message indicating that the filesystem checked contained *X* files using *Y* blocks leaving *Z* blocks free in the filesystem.

***** BOOT XENIX (NO SYNC!) *****

This is an advisory message indicating that a mounted filesystem or the root filesystem has been modified by *fsck*. If the UNIX system is not rebooted immediately without *sync*, the work done by *fsck* may be undone by the in-core copies of tables the UNIX system keeps. If the *-b* option of the *fsck* command was specified and the filesystem is *root*, a reboot is automatically done.

***** FILE SYSTEM WAS MODIFIED *****

This is an advisory message indicating that the current filesystem was modified by *fsck*.

fsdb

filesystem debugger

Syntax

/etc/fsdb special [-]

Description

fsdb can be used to patch up a damaged filesystem after a crash. It has conversions to translate block and i-numbers into their corresponding disk addresses. Also included are mnemonic offsets to access different parts of an i-node. These greatly simplify the process of correcting control block entries or descending the filesystem tree.

fsdb contains several error-checking routines to verify i-node and block addresses. These can be disabled if necessary by invoking *fsdb* with the optional - argument or by the use of the **O** symbol. (*fsdb* reads the i-size and f-size entries from the superblock of the filesystem as the basis for these checks.)

Numbers are considered decimal by default. Octal numbers must be prefixed with a zero. During any assignment operation, numbers are checked for a possible truncation error due to a size mismatch between source and destination.

fsdb reads a block at a time and will therefore work with raw as well as block I/O. A buffer management routine is used to retain commonly used blocks of data in order to reduce the number of read system calls. All assignment operations result in an immediate write-through of the corresponding block.

The symbols recognized by *fsdb* are:

#	absolute address
i	convert from i-number to i-node address
b	convert to block address
d	directory slot offset
+,-	address arithmetic
q	quit
>,<	save, restore an address
=	numerical assignment
=+	incremental assignment
=-	decremental assignment
="	character string assignment

O	error checking flip flop
p	general print facilities
f	file print facility
B	byte mode
W	word mode
D	double word mode
!	escape to shell

The print facilities generate a formatted output in various styles. The current address is normalized to an appropriate boundary before printing begins. It advances with the printing and is left at the address of the last item printed. The output can be terminated at any time by typing the delete character. If a number follows the **p** symbol, that many entries are printed. A check is made to detect block boundary overflows since logically sequential blocks are generally not physically sequential. If a count of zero is used, all entries to the end of the current block are printed. The print options available are:

i	print as i-nodes
d	print as directories
o	print as octal words
e	print as decimal words
c	print as characters
b	print as octal bytes

The **f** symbol is used to print data blocks associated with the current i-node. If followed by a number, that block of the file is printed. (Blocks are numbered from zero.) The desired print option letter follows the block number, if present, or the **f** symbol. This print facility works for small as well as large files. It checks for special devices and that the block pointers used to find the data are not zero.

Dots, tabs, and spaces may be used as function delimiters but are not necessary. A line with just a new-line character will increment the current address by the size of the data type last printed. That is, the address is set to the next byte, word, double word, directory entry or i-node, allowing the user to step through a region of a filesystem. Information is printed in a format appropriate to the data type. Bytes, words and double words are displayed with the octal address followed by the value in octal and decimal. A **.B** or **.D** is appended to the address for byte and double word values, respectively. Directories are printed as a directory slot offset followed by the decimal i-number and the character representation of the entry name. I-nodes are printed with labeled fields describing each element.

The following mnemonics are used for i-node examination and refer to the current working i-node:

md	mode
ln	link count

uid	user ID number
gid	group ID number
sz	file size
a#	data block numbers (0 - 12)
at	access time
mt	modification time
maj	major device number
min	minor device number

Examples

386i	prints i-number 386 in an i-node format. This now becomes the current working i-node.
ln=4	changes the link count for the working i-node to 4.
ln=+1	increments the link count by 1.
fc	prints, in ASCII, block zero of the file associated with the working i-node.
2i.fd	prints the first 32 directory entries for the root i-node of this filesystem.
d5i.fc	changes the current i-node to that associated with the 5th directory entry (numbered from zero) found from the above command. The first logical block of the file is then printed in ASCII.
512B.p0o	prints the superblock of this filesystem in octal.
2i.a0b.d7=3	changes the i-number for the seventh directory slot in the root directory to 3. This example also shows how several operations can be combined on one command line.
d7.nm="name"	changes the name field in the directory slot to the given string. Quotes are optional when used with **nm** if the first character is alphabetic.
a2b.p0d	prints the third block of the current i-node as directory entries.

Notes

The directory **/etc/fscmd.d/***TYPE* contains programs for each filesystem type; each of these programs applies some appropriate heuristic to determine whether the supplied *special* file is of the type for which it checks.

See Also

fsck(ADM), dir(F), filesystem(F), ''Patching a Filesystem with fsdb'' in the ''Using Filesystems'' chapter of the *System Administrator's Guide*

fsname

prints or changes the name of a file system

Syntax

fsname [-p] [**-s** name] /dev/*device*

Description

The */etc/fsname* utility is used to print or change the name of a filesystem. The options are:

-p Select the "pack" name field instead of the filesystem name field.

-s *name* Changes the specified field in the superblock.

The default action is to print the name of the filesystem.

See Also

mkfs(ADM), ustat(S), filesystem(F)

Value Added

fsname is an extension of AT&T System V provided in Altos UNIX System V.

fsphoto

performs periodic semi-automated system backups

Syntax

fsphoto [**-i**] schedule [drive]

Description

fsphoto, in conjunction with *fsave*(ADM), provides a semi-automated interface to *xbackup*(ADM) and *cpio*(C) for backing-up filesystems (*xbackup* can only be used to back up XENIX fileystems). A human operator is required to mount and dismount tapes or floppies at the appropriate times, so some interaction is necessary, but all such interaction is kept to a minimum to reduce the potential for human error.

The selection and timing of backups for all filesystems is governed by the *schedule*(ADM) database. The system administrator must set up this file, and make arrangements to run *fsphoto* on the implicitly defined schedule (normally once per weekday). *fsphoto* can be invoked most easily from the *sysadmsh*(ADM). *fsphoto* interprets *schedule*, and for each filesystem that should be backed-up on that day, runs *fsave* to interact with the operator and backup the filesystem without error.

The optional argument *drive* specifies the magtape or floppy device to use; the default is specified in the *schedule* file.

Backups may be postponed (via *fsave*) or interrupted. The resulting "partial" backups are automatically resumed the next time *fsphoto* is run: Any missed filesystems are backed-up as if the original backup had not been delayed. The **-i** flag ignores any pending partial backups.

If there is a pending partial backup, the normally scheduled backups are not done. This means that if a partial backup is resumed, and the normally scheduled backups are to be done, *fsphoto* must be run twice.

You must be the super-user to use this program.

Files

/usr/lib/sysadmin/schedule
Database describing which filesystems are to be backed-up when, and at what dump *level*.

/dev/tty
Source of interactive input.

/usr/lib/sysadmin/past
Record of filesystems successfully backed-up in the pending partial backup.

/tmp/backup$$
Temporary file for recording successfully backed-up filesystems.

See Also

fsave(ADM), schedule(ADM), xbackup(ADM), basename(C)

Diagnostics

fsphoto complains of syntax errors in *schedule*, and exits with a status of 1.

fsphoto complains about illegal or incorrect arguments, and exits with a status of 1.

An interrupt will cause an exit status of 2.

Notes

If a *drive* is explicitly given, the "raw" (**/dev/r***) form of the device should be used.

Value Added

fsphoto is an extension to AT&T System V developed in Altos UNIX System V.

fsstat

report file system status

Syntax

/etc/fsstat special_file

Description

The *fsstat* command 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. The *fsstat* command 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

filesystem(F)

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.

This command does not work on DOS filesystems.
The directory **/etc/fscmd.d/***TYPE* contains programs for each file system type, *fsstat* invokes the appropriate binary.

fstyp

determine file system identifier

Syntax

fstyp device

Description

The *fstyp* command 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*(S) and sometimes by *mount*(ADM) to mount file systems of different types.

fstyp runs the programs in **/etc/fscmd.d/***TYPE* in alphabetical order, passing *device* as an argument; if any program succeeds, its filesystem type identifier is printed and *fstyp* exits immediately. If no program succeeds, *fstyp* prints ''Unknown_fstyp'' to indicate failure.

See Also

mount(ADM), mount(S), sysfs(S)

fwtmp, wtmpfix

manipulate connect accounting records

Syntax

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

Description

fwtmp

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*(C), 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

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*(ADM) 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. *wtmpfix* uses these records to synchronize all time stamps in the file.

In addition to correcting time/date stamps, *wtmpfix* 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, *wtmpfix* reduces the chance that *acctcon*(ADM) will fail when processing connect accounting records.

Files

/etc/wtmp

See Also

acct(ADM), acctcms(ADM), acctcom(C), acctcon(ADM), acctmerg(ADM), acctprc(ADM), acctsh(ADM), ed(C), runacct(ADM), acct(S), acct(F), utmp(F)

Standards Conformance

fwtmp and *wtmpfix* are conformant with:

AT&T SVID Issue 2, Select Code 307-127.

goodpw

check a password for non-obviousness

Syntax

goodpw [**-absm**] [**-d** file] [**-r** reason] [**-MR** expr]

Description

goodpw reads from the standard input a proposed password and applies a variety of heuristic checks intended to spot poor password choices. These checks can include checking against user names, English words, and too short or too simple passwords. Which checks are applied depends on the settings in **/etc/default/goodpw**, the file specified by the **-d** option, and the expressions specified by the **-M** and **-R** options.

The first line read from the standard input is taken to be the proposed password. A list of ‘‘canonical forms’’ is then generated; the canonical form is the password sans any non-letters and with all letters converted to upper-case. The list always includes the canonical form of the password and may, depending on the settings in **/etc/default/goodpw**, also contain left or right ‘‘rotations’’ of the canonical form. A rotation to the left is a shifting of the second through last character one position to the left, with the first character becoming the last; a rotation to the right is similar but in the opposite direction. The canonical list so generated is what most of the checks are applied against; if any (possibly rotated) canonical form in the list fails a check, the password is considered inadvisable and is rejected.

Any subsequent lines read from the standard input are taken to be a ‘‘stop-list’’ of disallowed passwords. Each line in the stop-list is reduced to its canonical form and checked against the canonical list; if there is a match, the password is rejected.

When a password is rejected, the reason is written to the standard error output and *goodpw* exits with a non-zero status. If a password passes all checks and hence is not rejected, no message is issued and *goodpw* exits with a zero status.

The **-s** and **-m** options modify this behavior: If **-s** is specified, no reason is issued. If **-m** is specified, then:

1. the stop-list terminates with an empty line,

2. one line is written to the standard output indicating the acceptance or rejection of the password, and

3. the entire procedure is repeated using a new password and stop-list read from the standard input.

This allows one *goodpw* process to check multiple passwords. The line written by *goodpw* to the standard output if **-m** is specified is one of:

g The password passed all checks and seems to be acceptable.

r*reason*
The password was rejected for the indicated *reason*.

e*error*
The indicated system *error* occurred and it cannot be determined whether or not the password is acceptable.

If **-s** was specified, then no *reason* or *error* is written after a "**r**" or "**e**," respectively.

The other options are:

-a Use American spelling (default).

-b Use British spelling.

-r*reason*
Specify the message to be issued in case the proposed password matches one of those in the stop-list. The default *reason* is "same as previous password."

-d*file*
Read the named *file* (which should be in the same format as **/etc/default/goodpw**) and apply the various checks specified.

-M*expr*
The password must match *expr*, a boolean combination of regular expressions. If the first character of *expr* is a slash ("/") and a regular file by that name exists, the contents of that file are used as the expression. (If the file cannot be read, an error results.)

-R*expr*
The password must not match *expr*.

The boolean combination of regular expressions (*expr*) is built from the following operations:

$expr_1$ **&** $expr_2$
True if, and only if, both expressions $expr_1$ and $expr_2$ are true. If $expr_1$ is not true, $expr_2$ is not evaluated.

$expr_1$ | $expr_2$
True if either (or both) of $expr_1$ or $expr_2$ is true. If $expr_1$ is true, $expr_2$ is not evaluated.

$expr_1$ ^ $expr_2$
True if exactly one of $expr_1$ and $expr_2$ are true. Both $expr_1$ and $expr_2$ are always evaluated.

! *expr*
True if *expr* is not true; *expr* is always evaluated.

(*expr*)
True if, and only if, *expr* is true; *expr* is always evaluated.

/re/
True if, and only if, regular expression *re* matches the password. Any regular expression defined by *regcmp*(S) is understood; substrings defined by (...)$*n* are placed in "accumulator" *n*.

$*n* ~ */re/*
True if, and only if, accumulator *n* (**0**-**9**, or *****) matches regular expression *re*; accumulator star ("*") is the entire password.

$*n* !~ */re/*
True if, and only if, accumulator *n* is not matched by regular expression *re*.

The possible *goodpw* checks, their control settings in **/etc/default/goodpw**, and default values are:

MATCH=/usr/lib/goodpw/match
An expression (*expr*), or the name of file containing an expression, that the password must match. This expression also may be specified by the **-M** option.

REJECT=/usr/lib/goodpw/reject
An expression, or the name of a file containing an expression, that the password must not match. This expression may also be specified by the **-R** option.

LEFT_ROTATIONS=UNIQUE
How left rotations of the canonical form of the password should be treated: **NO** - ignored; **YES** - considered in other checks (i.e., added to the canonical list) and may contain duplications; **UNIQUE** - considered in other checks but must not contain any duplicates.

RIGHT_ROTATIONS=UNIQUE
Similarly for right rotations.

BOTH_ROTATIONS=UNIQUE
Similarly for rotations in both directions taken together.

AVOID_USERS=YES
Should the canonical list be checked against user login names and real names, obtained from **/etc/passwd**?

AVOID_GROUPS=YES
Should the canonical list be checked against group names and group member lists, obtained from **/etc/group**?

AVOID_MACHINES=YES
Should the canonical list be checked against machine names obtained from a number of files, including **/etc/systemid** and **/usr/lib/mail/top**?

AVOID_ALIASES=YES
Should the canonical list be checked against mail aliases obtained from **/usr/lib/mail/aliases**?

AVOID_WORDS=YES
Should the canonical list be checked for properly spelled English words?

BRITISH=NO
Should *spell* use American or British spelling? Which spelling to use may be specified by the **-a** and **-b** options.

SITECHECKS=NO
The name of a program to run to provide additional checking. The program is run with no arguments. Passed to the program on its standard input, on separate lines, is first the actual proposed password and then the canonical list. If the program exits with a non-zero status, the password is rejected.

SITEREASON=Rejected by site-specific check(s)
The reason to give when the **SITECHECKS** program rejects the password.

The values for the default settings can be adjusted to reflect the local system's security concerns. If **/etc/default/goodpw** does not exist or cannot be read, the above default values are used (except for **MATCH** and **REJECT**). The default **MATCH** expression matches any password which:

1. Contains lower-case letters, upper-case letters, and digits, and whose length is four or more characters; *or*,

2. Contains no lower-case letters, no upper-case letters, and no digits, and whose length is four or more characters; *or*,

3. Contains both lower-case letters and digits, or both upper-case letters and digits, or both lower- and upper-case letters, and whose length is five or more characters; *or*,

4. Contains nothing but lower-case letters, and whose length is six or more characters; *or*,

5. Contains nothing but upper-case latters, and whose length is six or more characters.

The default **REJECT** expression is:

/[Ss][Cc][Oo]/ | /[Xx][Ee][Nn][Ii][Xx]/

which matches any password that contains either "SCO" or "XENIX" regardless of case.

Files

/usr/lib/goodpw/match
Expression that all passwords must match; by default, it contains the above-described **MATCH** expression.

/usr/lib/goodpw/reject
Expression that no passwords should match; by default, it contains the above-described **REJECT** expression.

See Also

aliases(M), default(M), group(M), passwd(C), passwd(M), regex(S), spell(CT), systemid(M)

Notes

Not all valid English words are known to *spell*, and hence some English words are considered acceptable as passwords.

The maximum length of a password is 100 characters, none of which may be an ASCII NUL or LF (newline).

Empty passwords are always rejected.

Value Added

goodpw is an extension of AT&T System V provided in Altos UNIX System V.

graph

draws a graph

Syntax

graph [options]

Description

The *graph* command with no options takes pairs of numbers from the standard input as abscissas and ordinates of a graph. Successive points are connected by straight lines. The graph is encoded on the standard output for display by the *tplot*(ADM) filters.

If the coordinates of a point are followed by a non-numeric string, that string is printed as a label beginning on the point. Labels may be surrounded with quotes ", in which case they may be empty or contain blanks and numbers; labels never contain new-lines.

The following options are recognized, each as a separate argument:

-a	Supply abscissas automatically (they are missing from the input); spacing is given by the next argument (default 1). A second optional argument is the starting point for automatic abscissas (default 0 or lower limit given by **-x**).
-b	Break (disconnect) the graph after each label in the input.
-c	Character string given by next argument is default label for each point.
-g	Next argument is grid style, 0 no grid, 1 frame with ticks, 2 full grid (default).
-l	Next argument is label for graph.
-m	Next argument is mode (style) of connecting lines: 0 disconnected, 1 connected (default). Some devices give distinguishable line styles for other small integers (e.g., the Tektronix 4014: 2=dotted, 3=dash-dot, 4=short-dash, 5=long-dash).
-s	Save screen, do not erase before plotting.
-x [**l**]	If **l** is present, x axis is logarithmic. Next 1 (or 2) arguments are lower (and upper) x limits. Third argument, if present, is grid spacing on x axis. Normally these quantities are determined automatically.
-y [**l**]	Similarly for y.
-h	Next argument is fraction of space for height.
-w	Similarly for width.
-r	Next argument is fraction of space to move right before plotting.

-u Similarly to move up before plotting.

-t Transpose horizontal and vertical axes. (Option **-x** now applies to the vertical axis.)

A legend indicating grid range is produced with a grid unless the **-s** option is present. If a specified lower limit exceeds the upper limit, the axis is reversed.

See Also

graphics(ADM), tplot(ADM), spline(C)

Notes

The *graph* command stores all points internally and drops those for which there is no room.

Segments that run out of bounds are dropped, not windowed.

Logarithmic axes may not be reversed.

haltsys, reboot

closes out the file systems and shuts down the system

Syntax

/etc/haltsys [**-d**]
/etc/reboot

Description

The *haltsys* utility performs a *uadmin*( ) system call (see *uadmin*(S)) to flush out pending disk I/O, mark the file systems clean, and halt the processor. *haltsys* takes effect immediately, so user processes should be killed beforehand. *shutdown*(ADM) is recommended for normal system shutdown, since it warns users, terminates processes, then calls *haltsys*. Use *haltsys* directly only if you cannot run shutdown; for example, because of some system problem.

haltsys displays a prompt indicating that the system has been shut down and can be rebooted or powered down. If the **-d** option is used, the system will remain down and you are not given the option to reboot.

The *reboot* command performs the same function as *haltsys*, except the system is rebooted automatically afterwards.

Only the super-user can execute *haltsys* or *reboot*.

Notes

haltsys locks hard disk heads.

See Also

shutdn(S), uadmin(S), shutdown(ADM)

reboot was developed at the University of California, Berkeley, and is used with permission.

Value Added

haltsys and *reboot* are extensions of AT&T System V provided in Altos UNIX System V.

hdutil

hard disk utility for displaying and removing specific disk device names

Syntax

hdutil [**-v**] [[**-n** *device*] | [**-l** | **-d** | **-s** *drive_#*]]

Description

The *hdutil* utility helps you manage the device filenames associated with hard disks. Use *hdutil* to:

- Display a hard disk's device names and major/minor numbers.
- Display any device names that are part of a virtual disk.
- Remove a hard disk's device names.

The *hdutil* utility displays all the device names associated with a specified hard disk. *hdutil* can also display the major and minor numbers of the selected devices. It also lets you remove these device names (and delete references to them), thus allowing you to safely remove the hard disk. Once removed, the hard disk is inaccessible until the device nodes are added again with **mkdev hd**.

Options

-v Turn on verbose output. This causes the major and minor numbers for the displayed devices to be also shown. The **-v** option affects only options **-n** or **-s**.

-n *device*
Display all device files associated with the hard disk specified by the device name *device*.

The name *device* can be any device file found in the /**dev** directory. Although special devices (like **swap** or **recover**) can be used as the argument, the most useful form is **hd***dp*, where *d* is the disk "number" (really a letter), and *p* is the partition (e.g., **hda0**).

Disk numbers range from **a** to **z** and **A** to **Z**. Partition numbers range from **1** to a maximum of **4**. Device names in the alternate UNIX form (e.g., /**dev**/**dsk**/*dsp*) are not recognized. See *hd*(HW) for an explanation of device filename formats.

-l *drive_#*

Display any devices associated with this hard disk that are components of a virtual disk. The argument *drive_#* may be any currently used drive "number," which is really a letter ranging from **a** to **z** and **A** to **Z**.

-s *drive_#*

Display all device files associated with the hard disk specified by *drive_#*. The argument *drive_#* may be any currently used drive "number," which is really a letter ranging from **a** to **z** and **A** to **Z**.

-d *drive_#*

Delete all device nodes and file references to a hard disk specified by its *drive_#*. The argument *drive_#* may be any currently used drive "number," which is really a letter ranging from **b** to **z** and **A** to **Z**.

(Note that you cannot delete drive **a**, which is always the root drive since it is the first drive installed on every system.

Recall that the drive "number" letter represents the order in which you installed the drive, not its physical location. Thus, drive number **b** specifies the second drive added to the system; it might not be the drive in the second drive bay.

Use *scsinfo*(ADM) to determine the physical location of a specified hard disk number.

Notes

When a hard disk is removed with the **-d** option, all the device nodes associated with this disk in the **/dev**, **/dev/rdsk**, and the **/dev/dsk** directories are removed. All references to this disk in the following files are also removed: **/etc/checklist**, **/etc/default/filesys**, and **/usr/lib/mkdev/perms/HD**.

Note that since *hdutil* makes no changes to the kernel, it is possible to add the disk again without having to reboot. If the disk is to be added again, use the *mkdev hd* command. However, the disk might not be the same disk number (that is, **b**, **c**, **d**, ...) as it was originally if any other disks were added or deleted since its removal.

If the disk to be removed is part of a virtual disk or is currently mounted, *hdutil* will exit without removing any of its device nodes.

Files

/altos/bin/hdutil
/etc/default/filesys
/etc/checklist

/usr/lib/mkdev/perms/HD

See Also

mkdev(ADM), hd(HW), ''Adding Hard Disks'' and ''Virtual Disks'' in the *System Administrator's Guide*

Value Added

hdutil is an extension to AT&T System V provided in Altos UNIX System V.

id

print user and group IDs and names

Syntax

id

Description

The *id* command outputs the user and group IDs and the corresponding names of the invoking process. If the effective and real IDs are different, both are printed.

See Also

logname(C), getuid(S)

Standards Conformance

id is conformant with:

AT&T SVID Issue 2, Select Code 307-127;
and The X/Open Portability Guide II of January 1987.

idaddld

add or remove line disciplines from kernel configuration files

Syntax

/etc/conf/bin/idaddld [**-a** name routine1 ... routine8] [**-dc** name]

Description

idaddld is used to add or remove line discipline declarations from kernel configuration files. If no arguments are given then *idaddld* enters an interactive mode. In this mode the user can add, delete or view the current configuration. If a change is specified then the user is prompted to relink the kernel. If arguments are given on the command line then *idaddld* enters a non-interactive mode executing the specified command silently. It is the responsibility of the calling program to insure the kernel is relinked to effect the desired changes.

Options

The following options are available from the command line.

-a prefix routine1 ... routine8
Add a line discipline to configuration files. Prefix is a tag used to identify the line discipline for future inquiries or removal. For example, the terminal line discipline uses the prefix tty. Routine1 through routine8 define the list of line discipline routines. There must be eight routines with the keyword "nulldev" used as a placeholder. The order of the routines is critical. They must be ordered as follows:

open close read write ioctl rxint txint modemint

-d prefix
Remove the line discipline whose identifier matches prefix.

-c prefix
Scan the line discipline switch table for an entry which matches prefix. The program will exit with a return status 0 if a match is found and 1 otherwise.

Notes

When a line discipline is added, it is appended to the current switch table configuration.

Value Added

idaddld is an extension of AT&T System V provided in Altos UNIX System V.

idbuild, idmkenv, idmkunix, idconfig, idvidi, idscsi

build new UNIX system kernel

Syntax

/etc/conf/bin/idbuild

Description

This script builds a new UNIX system kernel using the current system configuration in **etc/conf/**. Kernel reconfigurations are usually done after a device driver is installed, or system tunable parameters are modified. The script uses the shell variable **$ROOT** from the user's environment as its starting path. Except for the special case of kernel development in a non-root source tree, the shell variable **ROOT** should always be set to null or to "/." *idbuild* exits with a return code of zero on success and non-zero on failure.

Building a new UNIX system image consists of generating new system configuration files, then link-editing the kernel and device driver object modules in the **etc/conf/pack.d** object tree. This is done by *idbuild* by calling the following commands:

etc/conf/bin/idconfig To build kernel configuration files.

etc/conf/bin/idmkunix To process the configuration files and link-edit a new UNIX system image.

The system configuration files are built by processing the Master and System files representing device driver and tunable parameter specifications. The files **etc/conf/cf.d/mdevice,** and **etc/conf/cf.d/mtune** represent the Master information. The file **etc/conf/cf.d/stune,** and the files specified in **etc/conf/sdevice.d/*** represent the System information. The kernel also has file system type information defined in the files specified by **etc/conf/sfsys.d/*** and **etc/conf/mfsys.d/***.

idvidi and *idscsi* read the video driver and SCSI driver configurations, respectively.

idconfig reads the system configuration files and reports any conflicts and errors.

idmkunix links the necessary modules to create the new kernel.

Once a new UNIX system kernel has been configured and linked, *idmkdenv* is invoked to back up the current **/unix** and replace it with the new kernel, and rebuild the kernel environment.

Diagnostics

Since *idbuild* calls other system commands to accomplish system reconfiguration and link editing, it will report all errors encountered by those commands, then clean up intermediate files created in the process. In general, the exit value 1 indicates an error was encountered by *idbuild*.

The errors encountered fall into the following categories:

Master file error messages.
System file error messages.
Tunable file error messages.
Compiler and Link-editor error messages.

All error messages are designed to be self-explanatory.

See Also

configure(ADM), idinstall(ADM), idtune(ADM), mdevice(F), mfsys(F), mtune(F), sdevice(F), sfsys(F), stune(F)

idcheck

returns selected information

Syntax

/etc/conf/bin/idcheck

Description

This command returns selected information about the system configuration. It is useful in add-on device Driver Software Package (DSP) installation scripts to determine if a particular device driver has already been installed, or to verify that a particular interrupt vector, I/O address or other selectable parameter is in fact available for use. The various forms are:

idcheck -p device-name [-i dir] [-r]

idcheck -v vector [-i dir] [-r]

idcheck -d dma-channel [-i dir] [-r]

idcheck -a -l lower_address -u upper_address [-i dir] [-r]

idcheck -c -l lower_address -u upper_address [-i dir] [-r]

This command scans the System and Master modules and returns:

100 if an error occurs.

0 if no conflict exists.

a positive number greater than 0 and less than 100 if a conflict exists.

The command line options are:

-r Report device name of any conflicting device on stdout.

-p *device-name* This option checks for the existence of four different components of the DSP. The exit code is the addition of the return codes from the four checks.

Add 1 to the exit code if the DSP directory under **/etc/conf/pack.d** exists.

Add 2 to the exit code if the Master module has been installed.

Add 4 to the exit code if the System module has been installed.

Add 8 to the exit code if the Kernel was built with the System module.

Add 16 to the exit code if a Driver.o is part of the DSP (vs. a stubs.c file).

-v *vector* — Returns "type" field of device that is using the vector specified (i.e., another DSP is already using the vector).

-d *dma-channel* — Returns 1 if the dma channel specified is being used.

-a — This option checks whether the IOA region bounded by "lower" and "upper" conflict with another DSP ("lower" and "upper" are specified with the **-l** and **-u** options). The exit code is the addition of two different return codes.

Add 1 to the exit code if the IOA region overlaps with another device.

Add 2 to the exit code if the IOA region overlaps with another device and that device has the 'O' option specified in the *type* field of the Master module. The 'O' option permits a driver to overlap the IOA region of another driver.

-c — Returns 1 if the CMA region bounded by "lower" and "upper" conflict with another DSP ("lower" and "upper" are specified with the **-l** and **-u** options).

-l *address* — Lower bound of address range specified in hex. The leading 0x is unnecessary.

-u *address* — Upper bound of address range specified in hex. The leading 0x is unnecessary.

-i *dir* — Specifies the directory in which the ID files *sdevice* and *mdevice* reside. The default directory is **/etc/conf/cf.d.**

Diagnostics

There are no error messages or checks for valid arguments to options. *idcheck* interprets these arguments using the rules of *scanf*(S) and queries the *sdevice* and *mdevice* files. For example, if a letter is used in the place of a digit, *scanf*(S) will translate the letter to 0. *idcheck* will then use this value in its query.

See Also

idinstall(ADM), mdevice(F), sdevice(F)

idinstall

add, delete, update, or get device driver configuration data

Syntax

/etc/conf/bin/idinstall -[**adug**] [**-e**] -[**msoptnirhcl**] dev_name

Description

The *idinstall* command is called by a Driver Software Package (DSP) Install script or Remove script to Add (**-a**), Delete (**-d**), Update (**-u**), or Get (**-g**) device driver configuration data. *idinstall* expects to find driver component files in the current directory. When components are installed or updated, they are moved or appended to files in the **/etc/conf** directory and then deleted from the current directory unless the **-k** flag is used. The options for the command are as follows:

Action Specifiers:

-a Add the DSP components

-d Remove the DSP components

-u Update the DSP components

-g Get the DSP components (print to std out, except Master)

Component Specifiers: (*)

-m Master component

-s System component

-o Driver.o component

-p Space.c component

-t Stubs.c component

-n Node (special file) component

-i Inittab component

-r Device Initialization (rc) component

-h Device shutdown (sd) component

-c Mfsys component: file system type config (Master) data

-l Sfsys component: file system type local (System) data

(*) If no component is specified, the default is all except for the **-g** option where a single component must be specified explicitly.

Miscellaneous:

-e Disable free disk space check

-k Keep files (do not remove from current directory) on add or update.

In the simplest case of installing a new DSP, the command syntax used by the DSP's Install dinstascript should be *idinstall -a dev_name*. In this case the command will require and install a Driver.o, Master and System entry, and optionally install the Space.c, Stubs.c, Node, Init, Rc, Shutdown, Mfsys, and Sfsys components if those modules are present in the current directory.

The **Driver.o**, **Space.c**, and **Stubs.c** files are moved to a directory in **/etc/conf/pack.d**. The *dev_name* is passed as an argument, which is used as the directory name. The remaining components are stored in the corresponding directories under **/etc/conf** in a file whose name is **dev_name**. For example, the Node file would be moved to **/etc/conf/node.d/dev_name.**

The *idinstall -m* usage provides an interface to the *idmaster* command which will add, delete, and update *mdevice* file entries using a Master file from the local directory. An interface is provided here so that driver writers have a consistent interface to install any DSP component.

As stated above, driver writers will generally use only the *idinstall -a dev_name* form of the command. Other options of *idinstall* are provided to allow an Update DSP (i.e., one that replaces an existing device driver component) to be installed, and to support installation of multiple controller boards of the same type.

If the call to *idinstall* uses the **-u** (update) option, it will:

overlay the files of the old DSP with the files of the new DSP.

invoke the *idmaster* command with the 'update' option if a Master module is part of the new DSP.

idinstall also does a verification that enough free disk space is available to start the reconfiguration process. This is done by calling the **idspace** command. *idinstall* will fail if insufficient space exists, and exit with a non-zero return code. The **-e** option bypasses this check.

idinstall makes a record of the last device installed in a file (**/etc/.last_dev_add**), and saves all removed files from the last delete operation in a directory (**/etc/.last_dev_del**). These files are recovered by */etc/conf/bin/idmkenv* whenever it is determined that a system reconfiguration was aborted due to a power failure or unexpected system reboot.

Diagnostics

An exit value of zero indicates success. If an error was encountered, *idinstall* will exit with a non-zero value, and report an error message. All error messages are designed to be self-explanatory. Typical error message that can be generated by *idinstall* are as follows:

Device package already exists.
Cannot make the driver package directory.
Cannot remove driver package directory.
Local directory does not contain a Driver object (Driver.o) file.
Local directory does not contain a Master file.
Local directory does not contain a System file.
Cannot remove driver entry.

See Also

idspace(ADM), idcheck(ADM), mdevice(F), sdevice(F)

idleout

logs out idle users

Syntax

idleout [minutes I hours:minutes]

Description

The *idleout* command monitors line activity and logs out users whose terminal remains idle longer than a specified period of time. Minutes are assumed; if a colon appears in the number, hours are assumed.

The utility uses a default file, */etc/default/idleout*, to indicate the number of hours a user's terminal may remain idle before being logged out. This file has one entry:

IDLETIME=*time*

The time format is identical to that used on the command line. The time specified in the default file is overridden by *idletime* if *idletime* is specified on the command line. Note that, if *idletime* is zero, no monitoring takes place and idle users are not logged out. You can either run *idleout* ffrom the command line, or, to have continuous coverage, you must add the program name in **/etc/rc.d/8/userdef** to see to it that the program is run each time the system is rebooted.

Files

/etc/default/idleout
/etc/utmp
/etc/wtmp

See Also

who(C), getut(S), kill(S)

Value Added

idleout is an extension of AT&T System V provided in Altos UNIX System V.

idmemtune

adjusts tunable parameters to match system memory

Syntax

/etc/conf/bin/idmemtune [**-f**] [**-r** *root*] [**-m** *memtune*]
[**-M** *memsize*] [**-b** *basemem*]

Description

The *idmemtune* command uses the **memtune** file to interactively configure tunable parameters to system memory size. Before adjusting each parameter, the user is prompted to either accept (by entering **y**) or reject the proposed change. Each parameter is then configured using **idtune**(ADM).

Options

-f Suppress the interactive mode and set tunable parameters silently.

-r *root*
Use *root* as an alternate root directory, below which other absolute pathnames are derived. This option overrides the ROOT environment variable.

-m *memtune*
Use *memtune* as an alternate **memtune** file (the default is *root*/**etc/conf.d/memtune**).

-M *memsize*
Adjust tunable parameters to match *memsize* instead of the actual system memory size.

-b *basemem*
Use the value *basemem* megabytes as the nase memory size corresponding to the base values in the **memtune** file instead of 8 megabytes.

Files

/etc/conf/cf.d/memtune
/etc/conf/cf.d/stune

See Also

idtune(ADM), uconfig(ADM), memtune(F), stune(F)

Value Added

idmemtune(ADM) is an extension to AT&T System V provided in Altos UNIX System V.

idmkinit

read files containing specifications

Syntax

/etc/conf/bin/idmkinit

Description

This command reads the files containing specifications of **/etc/inittab** entries from **/etc/conf/init.d** and constructs a new **inittab** file in **/etc/conf/cf.d**. It returns 0 on success and a positive number on error.

The files in **/etc/conf/init.d** are copies of the Init modules in device Driver Software Packages (DSP). There is at most one Init file per DSP. Each file contains one line for each **inittab** entry to be installed. There may be multiple lines (i.e., multiple **inittab** entries) per file. An **inittab** entry has the form (the *id* field is often called the *tag*):

id:rstate:action:process

The Init module entry must have one of the following forms:

action:process

rstate:action:process

id:rstate:action:process

When *idmkinit* encounters an entry of the first type, a valid **id** field will be generated, and an **rstate** field of 2 (indicating run on init state 2) will be generated. When an entry of the second type is encountered, only the **id** field is prefixed. An entry of the third type is incorporated into the new **inittab** unchanged.

Since add-on **inittab** entries specify init state 2 for their **rstate** field most often, an entry of the first type should almost always be used. An entry of the second type may be specified if you need to specify other than state 2. DSP's should avoid specifying the **id** field as in the third entry since other add-on applications or DSPs may have already used the **id** value you have chosen. The **/etc/init** program will encounter serious errors if one or more **inittab** entries contain the same **id** field.

idmkinit determines which of the three forms above is being used for the entry by requiring each entry to have a valid **action** keyword. Valid **action** values are as follows:

off
respawn
ondemand
once
wait
boot
bootwait
powerfail
powerwait
initdefault
sysinit

The *idmkinit* command is called automatically upon entering init state 2 on the next system reboot after a kernel reconfiguration to establish the correct **/etc/inittab** for the running **/unix** kernel. *idmkinit* can be called as a user level command to test modification of **inittab** before a DSP is actually built. It is also useful in installation scripts that do not reconfigure the kernel but need to create **inittab** entries. In this case, the **inittab** generated by *idmkinit* must be copied to **/etc/inittab**, and a *telinit q* command must be run to make the new entry take effect.

The command line options are

-o *directory* **inittab** will be created in the directory specified rather than **/etc/conf/cf.d**.

-i *directory* The ID file **init.base**, which normally resides in **/etc/conf/cf.d**, can be found in the directory specified.

-e *directory* The Init modules that are usually in **/etc/conf/init.d** can be found in the directory specified.

-# Print debugging information.

Diagnostics

An exit value of zero indicates success. If an error was encountered, *idmkinit* will exit with a non-zero value and report an error message. All error messages are designed to be self-explanatory.

See Also

idbuild(ADM), idinstall(ADM), idmknod(ADM), init(M), inittab(F)

idmknod

removes nodes and reads specifications of nodes

Syntax

/etc/conf/bin/idmknod

Description

This command performs the following functions:

- Removes the nodes for non-required devices (those that do not have an 'r' in field 3 of the device's *mdevice* entry) from **/dev**. Ordinary files will not be removed. If the **/dev** directory contains subdirectories, those subdirectories will be transversed and nodes found for non-required devices will be removed as well. If empty subdirectories result due to the removal of nodes, the subdirectories are then removed.

- Reads the specifications of nodes given in the files contained in **/etc/conf/node.d** and installs these nodes in **/dev**. If the node specification defines a path containing subdirectories, the subdirectories will be made automatically.

- Returns 0 on success and a positive number on error.

The *idmknod* command is run automatically upon entering init state 2 on the next system reboot after a kernel reconfiguration to establish the correct representation of device nodes in the **/dev** directory for the running **/unix** kernel. *idmknod* can be called as a user level command to test modification of the **/dev** directory before a DSP is actually built. It is also useful in installation scripts that do not reconfigure the kernel, but need to create **/dev** entries.

The files in **/etc/conf/node.d** are copies of the *Node* modules installed by device Driver Software Packages (DSP). There is at most one file per DSP. Each file contains one line for each node that is to be installed. The format of each line is:

Name of device entry (field 1) in the *mdevice* file (The *mdevice* entry will be the line installed by the DSP from its *Master* module). This field must be from 1 to 8 characters in length. The first character must be a letter. The others may be letters, digits, or underscores.

Name of node to be inserted in **/dev**. The first character must be a letter. The others may be letters, digits, or underscores. This field can be a path relative to **/dev**, and *idmknod* will create subdirectories as needed.

The character **b** or **c**. A **b** indicates that the node is a 'block' type device and **c** indicates 'character' type device.

Minor device number. This value must be between 0 and 255. If this field is a non-numeric, it is assumed to be a request for a streams clone device node, and *idmknod* will set the minor number to the value of the major number of the device specified.

Some example node file entries are as follows:

asy tty00 c 1
makes /dev/tty00 for device 'asy' using minor device 1.

qt rmt/c0s0 c 4
makes /dev/rmt/c0s0 for device 'qt' using minor device 4.

clone net/nau/clone c nau
makes /dev/net/nau/clone for device 'clone'. The minor device number is set to the major device number of device 'nau'.

The command line options are:

-o *directory* Nodes will be installed in the directory specified rather than **/dev**.

-i *directory* The file *mdevice* which normally resides in **/etc/conf/cf.d**, can be found in the directory specified.

-e *directory* The *Node* modules that normally reside in **/etc/conf/node.d** can be found in the directory specified.

-s Suppress removing nodes (just add new nodes).

Diagnostics

An exit value of zero indicates success. If an error was encountered due to a syntax or format error in a *node* entry, an advisory message will be printed to *stdout* and the command will continue. If a serious error is encountered (i.e., a required file cannot be found), *idmknod* will exit with a non-zero value and report an error message. All error messages are designed to be self-explanatory.

See Also

idinstall(ADM), idmkinit(ADM), mdevice(F), sdevice(F)

idspace

investigates free space

Syntax

/etc/conf/bin/idspace [**-i** inodes] [**-r** blocks] [**-u** blocks]
[**-t** blocks]

Description

This command investigates free space in /, **/usr**, and **/tmp** filesystems to determine whether sufficient disk blocks and inodes exist in each of potentially 3 filesystems. The default tests that *idspace* performs are as follows:

- Verify that the root filesystem (/) has 400 blocks more than the size of the current **/unix**. This verifies that a device driver being added to the current **/unix** can be built and placed in the root directory. A check is also made to insure that 100 inodes exist in the root directory.
- Determine whether a **/usr** filesystem exists. If it does exist, a test is made that 400 free blocks and 100 inodes are available in that filesystem. If the filesystem does not exist, *idspace* does not complain since files created in **/usr** by the reconfiguration process will be created in the root filesystem and space requirements are covered by the test above.
- Determine whether a **/tmp** filesystem exists. If it does exist, a test is made that 400 free blocks and 100 inodes are available in that filesystem. If the filesystem does not exist, *idspace* does not complain since files created in **/tmp** by the reconfiguration process will be created in the root filesystem and space requirements are covered by the test above.

The command line options are:

-i *inodes* This option overrides the default test for 100 inode in all of the *idspace* checks.

-r *blocks* This option overrides the default test for **/unix** size + 400 blocks when checking the root (/) filesystem. When the **-r** option is used, the **/usr** and **/tmp** filesystems are not tested unless explicitly specified.

-u *blocks* This option overrides the default test for 400 blocks when checking the **/usr** filesystem. When the **-u** option is used, the root (/) and **/tmp** filesystems are not tested unless explicitly specified. If **/usr** is not a separate filesystem, an error is reported.

-t *blocks* This option overrides the default test for 400 blocks when checking the **/tmp** filesystem. When the **-t** option is used, the root (/) and **/usr** filesystems are not tested unless explicitly specified. If **/tmp** is not a separate filesystem, an error is reported.

Diagnostics

An exit value of zero indicates success. If insufficient space exists in a filesystem or an error was encountered due to a syntax or format error, *idspace* will report a message. All error messages are designed to be self-explanatory. The specific exit values are as follows:

0 success.

1 command syntax error, or needed file does not exist.

2 filesystem has insufficient space or inodes.

3 requested filesystem does not exist (**-u** and **-t** options only).

See Also

idbuild(ADM), idinstall(ADM)

idtune

attempts to set value of a tunable parameter

Syntax

/etc/conf/bin/idtune [**-f** | **-m**] name value

Description

This script attempts to set the value of a tunable parameter. The tunable parameter to be changed is indicated by *name*. The desired value for the tunable parameter is *value*.

If there is already a value for this parameter (in the **stune** file), the user will normally be asked to confirm the change with the following message:

```
Tunable Parameter name is currently set to old_value.
Is it OK to change it to value? (y/n)
```

If the user answers **y**, the change will be made. Otherwise, the tunable parameter will not be changed, and the following message will be displayed:

```
name left at old_value.
```

However, if the **-f** (force) option is used, the change will always be made and no messages will ever be given.

If the **-m** (minimum) option is used and there is an existing value which is greater than the desired value, no change will be made and no message will be given.

If system tunable parameters are being modified as part of a device driver or application add-on package, it may not be desirable to prompt the user with the above question. The add-on package Install script may chose to override the existing value using the **-f** or **-m** options. However, care must be taken not to invalidate a tunable parameter modified earlier by the user or another add-on package.

In order for the change in parameter to become effective, the UNIX system kernel must be rebuilt and the system rebooted.

Diagnostics

The exit status will ne non-zero if errors are encountered.

See Also

idbuild(ADM), mtune(F), stune(F)

infocmp

compare or print out terminfo descriptions

Syntax

infocmp [**-d**] [**-c**] [**-n**] [**-I**] [**-L**] [**-C**] [**-r**] [**-u**] [**-s d**|**i**|**l**|**c**] [**-v**] [**-V**] [**-1**] [**-w** width] [**-A** directory] [**-B** directory] [termname ...]

Description

The *infocmp* command can be used to compare a binary *terminfo*(F) entry with other terminfo entries, rewrite a *terminfo*(F) description to take advantage of the **use=** terminfo field, or print out a *terminfo*(F) description from the binary file [*term*(F)] 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]

The *infocmp* command compares the *terminfo*(F) 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, **-1** 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.

-I use the *terminfo*(F) 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. *infocmp* 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*(F), but which are derivable from other *terminfo*(F) variables, will be output. Not all *terminfo*(F) 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*(F) 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*(F) format will not necessarily reproduce the original *terminfo*(F) source.

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

Terminfo	Termcap	Representative Terminals
%p1%c	%.	adm
%p1%d	%d	hp, ANSI standard, vt100
%p1%'x'%+%c	%+x	concept
%i	%i	ANSI standard, vt100
%p1%?%'x'%>%t%p1%'y'%+%;	%>xy	concept
%p2 is printed before %p1	%r	hp

Use= Option [-u]

-u produce a *terminfo*(F) 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*(C) 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. *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 d|i|l|c] [-v] [-V] [-1] [-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* data base.
- **i** sort by *terminfo* name.
- **l** sort by the long C variable name.
- **c** sort 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 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.

Changing Data Bases [-A directory] [-B directory]

The location of the compiled *terminfo*(F) data base 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*(F) data base, 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 data bases. 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*(F) data base for a comparison to be made.

Files

/usr/lib/terminfo/?/* compiled terminal description data base

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./fl
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./fl
The **-u**, **-d**, and **-c** options require at least two terminal names.

See Also

captoinfo(ADM), tic(C), curses(S), term(F), terminfo(F)

initcond

special security actions for init and getty

Syntax

initcond [**init** | **getty**] [args ...]

Description

To save space in the *init*(M) and *getty*(M) programs, which are memory resident, the space intensive security actions are done in *initcond* as a sub-process of these programs.

If the argument is **init,** one of two actions may occur. First, no argument means that *initcond* should prompt for and verify a single user password if required by the System Default database. This is used for password checking before a single user shell. Second, if two other arguments are supplied, they are the terminal device name and the user name respectively of the session that just terminated. This information is reflected in both the Protected Password and Terminal Control databases.

If the argument is **getty**, and one additional argument is provided, it is the terminal to be invalidated before a login. *initcond* invalidates a terminal by setting a restricted set of permissions on the terminal device and by using *stopio*(S) to invalidate all open file descriptors that reference the terminal. These include synonym devices for the same physical device as listed in the device assignment database.

Files

/tcb/files/initcondlog - Log file for *init* and *getty* events
/etc/auth/system/ttys - Terminal Control database
/etc/auth/system/devassign - Device Assignment database

See Also

getprtcent(S), stopio(S), getdvagent(S), ''Maintaining System Security,'' chapter of the *System Administrator's Guide*

Value Added

initcond is an extension of AT&T System V provided in Altos UNIX System V.

initscript

defines environment for programs executed by init(M)

Description

/etc/initscript is used to define the default environment for all commands started by *init*(M). It is the central location for environment variables that must be defined for a command. This avoids the problems inherent in hard-coded values for such variables as HZ and PATH. The following is the default contents of **/etc/initscript**:

```
PATH=/bin:/usr/bin
export PATH

HZ=60
export HZ

[ -x /etc/TIMEZONE ] && . /etc/TIMEZONE

ulimit 1000

eval exec "$4"
```

Notes

Variables other than TZ should not be defined in **/etc/TIMEZONE** as in previous releases; they should be moved to **/etc/initscript**.

Value Added

initscript is an extension of AT&T System V provided in Altos UNIX System V.

install

install commands

Syntax

/etc/install [**-c** dira] [**-f** dirb] [**-i**] [**-n** dirc] [**-m** mode] [**-u** user]
[**-g** group] [**-o**] [**-s**] file [dirx ...]

Description

The *install* command is most commonly used in "makefiles" [see *make*(CP)] to install a *file* (updated target file) in a specific place within a file system. Each *file* is installed by copying it into the appropriate directory, thereby retaining the mode and owner of the original command. The program prints messages telling the user exactly what files it is replacing or creating and where they are going.

If no options or directories (*dirx* ...) are given, *install* will search a set of default directories (**/bin**, **/usr/bin**, **/etc**, **/lib**, and **/usr/lib**, in that order) for a file with the same name as *file*. When the first occurrence is found, *install* issues a message saying that it is overwriting that file with *file*, and proceeds to do so. If the file is not found, the program states this and exits without further action.

If one or more directories (*dirx* ...) are specified after *file*, those directories will be searched before the directories specified in the default list.

The meanings of the options are:

-c *dira* Installs a new command (*file*) in the directory specified by *dira*, only if it is not found. If it is found, *install* issues a message saying that the file already exists, and exits without overwriting it. May be used alone or with the **-s** option.

-f *dirb* Forces *file* to be installed in given directory, whether or not one already exists. If the file being installed does not already exist, the mode and owner of the new file will be set to **755** and **bin**, respectively. If the file already exists, the mode and owner will be that of the already existing file. May be used alone or with the **-o** or **-s** options.

-i Ignores default directory list, searching only through the given directories (*dirx* ...). May be used alone or with any other options except **-c** and **-f**.

-n *dirc* If *file* is not found in any of the searched directories, it is put in the directory specified in *dirc*. The mode and owner of the new file will be set to **755** and **bin**, respectively. May be used alone or with any other options except **-c** and **-f**.

-m *mode* The mode of the new file is set to *mode*. Only available to the superuser.

-u *user* The owner of the new file is set to *user*. Only available to the superuser.

-g *group* The group id of the new file is set to *group*. Only available to the superuser.

-o If *file* is found, this option saves the "found" file by copying it to **OLD***file* in the directory in which it was found. This option is useful when installing a frequently used file such as */bin/sh* or */etc/getty*, where the existing file cannot be removed. May be used alone or with any other options except **-c**.

-s Suppresses printing of messages other than error messages. May be used alone or with any other options.

See Also

make(CP)

installpkg

install package

Syntax

installpkg

Description

The *installpkg* command is used to install an AT&T-style UNIX system software package.

You will have to be *root* to install certain packages successfully.

You will be prompted to insert the floppy disk that the installation package resides on. Everything else is automatic.

Notes

You must invoke *installpkg* on the console.

This command does not work on packages installed with **custom**(ADM).

See Also

displaypkg(ADM), removepkg(ADM)

integrity

examine system files against the authentication database

Syntax

integrity [**-v**] [**-e**] [**-m**]

Description

integrity traverses the File Control database and compares each entry in turn to the real file in the file system. If the owner, group or permissions are different, an error message is output.

Wild card entries in the File Control database are handled as follows. For file names, those file names that have /* as the last entry are treated as wild cards. Any file in the directory matches that entry, unless the specific file under consideration has its own (non-wild card) entry in the database appearing before the wild card entry. In this case, the file is ignored in the check because it would have already been located previously. For owners (groups), if the File Control entry does not explicitly list an owner (group), all owners (groups) match correctly.

The **-v** option lists all files under consideration, even those that match. The **-e** option explains why discretionary checks fail and exactly what the discrepancy is.

Normally, (non-wild card type) files in the File Control database that are missing from the file system are not reported. The **-m** option will override that default and report such missing files.

Files

/etc/auth/system/files - File Control database
/etc/auth/system/default - System Defaults database

See Also

authck(ADM), stat(S), getprfient(S), "Maintaining System Security," chapter of the *System Administrator's Guide*

Diagnostics

integrity returns a zero exit status if there are no discrepancies. Otherwise, *integrity* returns a positive value equal to the number of discrepancies.

Value Added

integrity is an extension of AT&T System V provided in Altos UNIX System V.

ipcrm

removes a message queue, semaphore set or shared memory ID

Syntax

ipcrm [options]

Description

ipcrm removes one or more specified messages, a semaphore or shared memory identifiers. The identifiers are specified by the following *options*:

-q *msqid* removes the message queue identifier *msqid* from the system and destroys the message queue and data structure associated with it.

-m *shmid* removes the shared memory identifier *shmid* from the system. The shared memory segment and data structure associated with it are destroyed after the last detach.

-s *semid* removes the semaphore identifier *semid* from the system and destroys the set of semaphores and data structure associated with it.

-Q *msgkey* removes the message queue identifier, created with key *msgkey*, from the system and destroys the message queue and data structure associated with it.

-M *shmkey* removes the shared memory identifier, created with key *shmkey*, from the system. The shared memory segment and data structure associated with it are destroyed after the last detach.

-S *semkey* removes the semaphore identifier, created with key *semkey*, from the system and destroys the set of semaphores and data structure associated with it.

The details of the removes are described in *msgctl*(S), *shmctl*(S), and *semctl*(S). The identifiers and keys may be found by using *ipcs*(ADM).

See Also

ipcs(ADM), msgctl(S), msgget(S), msgop(S), semctl(S), semget(S), semop(S), shmctl(S), shmget(S), shmop(S)

Note

ipcrm cannot be used to remove semaphores created using *creatsem*(S) or to remove shared memory created using *sdget*(S).

ipcs

reports the status of inter-process communication facilities

Syntax

ipcs [options]

Description

ipcs prints certain information about active inter-process communication facilities. Without *options*, information is printed in short format for message queues, shared memory, and semaphores that are currently active in the system. Otherwise, the information that is displayed is controlled by the following *options*:

-q Print information about active message queues.
-m Print information about active shared memory segments.
-s Print information about active semaphores.

If any of the options **-q**, **-m**, or **-s** are specified, information about only those indicated are displayed. If none of the three options are specified, information about all three are displayed.

-b Print biggest allowable size information (maximum number of bytes in messages on queue for message queues, size of segments for shared memory, and number of semaphores in each set for semaphores). See below, for the meaning of columns in a listing.

-c Print creator's login name and group name. See below.

-o Display information on outstanding usage (number of messages on queue, total number of bytes in messages on queue, and the number of processes attached to shared memory segments).

-p Display process number information. (Process ID of last process to send a message and process ID of last process to receive a message on message queues. It displays the process ID of the creating process and the process ID of the last process to attach or detach on shared memory segments.) See below.

-t Print time information. (Time of the last control operation that changed the access permissions for all facilities. Time of last *msgsnd* and last *msgrcv* on message queues, last *shmat* and last *shmdt* on shared memory, and last *semop*(S) on semaphores.) See below.

-a Use all print *options*. (This is a shorthand notation for **-b**, **-c**, **-o**, **-p**, and **-t**.)

-C *corefile*
Use the file *corefile* in place of **/dev/kmem**.

-N *namelist*
The argument will be taken as the name of an alternate *namelist* (**/unix** is the default).

-X Print information about XENIX interprocess communication, in addition to the standard interprocess communication status. The XENIX process information describes a second set of semaphores and shared memory. Note that the **-p** option does not print process number information for XENIX shared memory, and the **-t** option does not print time information about XENIX semaphores and shared memory.

The column headings and the meaning of the columns in an *ipcs* listing are given below; the letters in parentheses indicate the *options* that cause the corresponding heading to appear; **all** means that the heading always appears. Note that these *options* only determine what information is provided for each facility; they do *not* determine which facilities will be listed.

T (all) Type of the facility:

- **q** message queue;
- **m** shared memory segment;
- **s** semaphore.

ID (all) The identifier for the facility entry. Note that ID is "X" for facilities created using *creatsem*(S) or *sdget*(S).

KEY (all) The key used as an argument to *msgget*, *semget*, or *shmget* to create the facility entry. (Note: The key of a shared memory segment is changed to **IPC_PRIVATE** from when the segment has been removed until all processes attached to the segment detach it.)

MODE (all) The facility access modes and flags: The mode consists of 11 characters that are interpreted as follows:
The first two characters are:

- **R** if a process is waiting on a *msgrcv*;
- **S** if a process is waiting on a *msgsnd*;
- **D** if the associated shared memory segment has been removed. It will disappear when the last process attached to the segment detaches it;
- **C** if the associated shared memory segment is to be cleared when the first attach is executed;
- **-** if the corresponding special flag is not set.

The next 9 characters are interpreted as three sets of three bits each. The first set refers to the owner's permissions; the next to permissions of others in the user-group of the facility entry; and the last to all others. Within each set, the first character indicates permission to read, the second character indicates permission to write or alter the facility entry, and the last character is currently unused.

The permissions are indicated as follows:

r	if read permission is granted;
w	if write permission is granted;
a	if alter permission is granted;
-	if the indicated permission is *not* granted.

OWNER (all)	The login name of the owner of the facility entry.
GROUP (all)	The group name of the group of the owner of the facility entry.
CREATOR(a,c)	The login name of the creator of the facility entry.
CGROUP(a,c)	The group name of the group of the creator of the facility entry.
CBYTES (a,o)	The number of bytes in messages currently outstanding on the associated message queue.
QNUM (a,o)	The number of messages currently outstanding on the associated message queue.
QBYTES (a,b)	The maximum number of bytes allowed in messages outstanding on the associated message queue.
LSPID (a,p)	The process ID of the last process to send a message to the associated queue.
LRPID (a,p)	The process ID of the last process to receive a message from the associated queue.
STIME (a,t)	The time the last message was sent to the associated queue.
RTIME (a,t)	The time the last message was received from the associated queue.
CTIME (a,t)	The time when the associated entry was created or changed.
NATTCH (a,o)	The number of processes attached to the associated shared memory segment.
SEGSZ (a,b)	The size of the associated shared memory segment.
CPID (a,p)	The process ID of the creator of the shared memory entry.
LPID (a,p)	The process ID of the last process to attach or detach the shared memory segment.
ATIME (a,t)	The time the last attach was completed to the associated shared memory segment.
DTIME (a,t)	The time the last detach was completed on the associated shared memory segment.
NSEMS (a,b)	The number of semaphores in the set associated with the semaphore entry.
OTIME (a,t)	The time the last semaphore operation was completed on the set associated with the semaphore entry.

Files

/unix	system namelist
/dev/kmem	memory
/etc/passwd	user names
/etc/group	group names

See Also

msgop(S), semop(S), shmop(S) in the *Programmer's Reference Manual*

Warning

If the user specifies either the -C or -N flag, the real and effective UID/GID will be set to the real UID/GID of the user invoking *ipcs*.

Notes

Things can change while *ipcs* is running; the picture it gives is only a close approximation.

Authorization

The behavior of this utility is affected by assignment of the **mem** authorization, which is usually reserved for system administrators. If you do not have this authorization, the output will be restricted to data pertaining to your activities only. Refer to the "Using a Trusted System" chapter of the *User's Guide* for more details.

kbmode

set keyboard mode or test keyboard support

Syntax

kbmode command [file]

Description

This command can be used to determine if your system keyboard supports AT mode. If it does, this utility can change the keyboard mode in between AT mode and PC/XT compatibility mode.

If the **file** argument is specified, it should be a tty device of one of the multiscreens of the keyboard's group.

Valid commands are:

test	- determine if keyboard supports AT mode
at	- set keyboard to AT mode
xt	- set keyboard to PC/XT compatibility mode

Notes

Some keyboards look like AT keyboard but do not support AT mode. Setting such a keyboard to AT mode will render it useless, unless it can be set to XT mode from another (serial) terminal.

See Also

keyboard(HW)

Value Added

kbmode is an extension of AT&T System V provided in Altos UNIX System V.

killall

kill all active processes

Syntax

/etc/killall [signal]

Description

The *killall* command is used by **/etc/shutdown** to kill all active processes not directly related to the shutdown procedure.

The *killall* command terminates all processes with open files so that the mounted file systems will be unbusied and can be unmounted.

The *killall* command sends *signal* [see *kill*(C)] to all processes not belonging to the above group of exclusions. If no *signal* is specified, a default of **9** is used.

Files

/etc/shutdown

See Also

kill(C), ps(C), shutdown(ADM), signal(S)

Notes

The *killall* command can be run only by the super-user.

Standards Conformance

killall is conformant with:

AT&T SVID Issue 2, Select Code 307-127.

labelit

provide labels for filesystems

Syntax

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

Description

The *labelit* command can be used to provide labels for unmounted disk file systems or file systems being copied to tape. 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/0s3**). 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 super block.

See Also

sh(C), filesystem(F)

Standards Conformance

labelit is conformant with:

AT&T SVID Issue 2, Select Code 307-127.

link, unlink

link and unlink files and directories

Syntax

/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*(C) and *rmdir*(C) commands be used instead of the *unlink* command.

The only difference between *ln*(C) and *link*/*unlink* is that the latter do exactly what they are told to do, abandoning all error checking. This is because they directly invoke the *link*(S) and *unlink*(S) system calls.

See Also

rm(C), link(S), unlink(S)

Notes

These commands can be run only by the super-user.

Standards Conformance

link and *unlink* are conformant with:

AT&T SVID Issue 2, Select Code 307-127;
The X/Open Portability Guide II of January 1987;
IEEE POSIX Std 1003.1-1988 with C Standard Language-Dependent System Support;
and NIST FIPS 151-1.

link_unix

builds a new UNIX system kernel

Syntax

/etc/conf/cf.d/link_unix

Description

After installing a device driver, use *link_unix* to build a new UNIX system kernel. This script builds **/etc/conf/cf.d/unix** using the current system configuration in **/etc/conf.**

See Also

configure(ADM), idbuild(ADM), "Adding Device Drivers with the Link Kit" in the *System Administrator's Guide*

Value Added

link_unix is an extension of AT&T System V provided in Altos UNIX System V.

list

list processor channel for MMDF

Syntax

list

Description

List is an MMDF channel program for handling mailing lists. The channel functions as a feed-through between *deliver* and *submit*. The list channel has its own host table and domain table with one entry for the pseudo host ''list-processor'' or something similar. This program is called by the program *deliver* and is not meant to be invoked by users directly.

The *list* channel performs two basic services. First, it postpones the verification of the list addresses and performs the (possibly lengthy) verification in the background when the *list* channel resubmits the message to the mail system. This prevents tying up a network connection or a user's terminal when verifying a long mailing list. Second, the *list* channel will, under special circumstances, change the return address for the message to a generic maintainer's address. The return address is determined by first taking the destination address (e.g. ''largelist'') and seeing if there is an address in the alias file called ''largelist-request''. If there is, then ''largelist-request'' is used as the return address. If that was not found, the list channel checks to see if the destination address has a trailing ''-outbound''. If so, this is stripped and a ''-request'' is added and the lookup in the alias file is made a second time. If the ''-request'' address is found, then that address is used as the return address. If no ''-request'' address is found, then the original return address is used (normally the address of the sender).

To use the *list* channel to process a list, it is generally necessary to make three entries in the alias file(s). Let us say that we wish to set up a list called ''largelist'' and we want this list to be processed by the *list* channel. We would need the following entries in the alias file:

largelist:	largelist-outbound@list-processor
largelist-outbound:	</usr/mmdf/lists/largelist-file
largelist-request:	maintainer

The first line causes mail sent to ''largelist'' to be sent through the list processor, readdressed to ''largelist-outbound''. The second line is what actually references the mailing list file for ''largelist''. The third line is optional, and is used to set up the (informal) standard maintenance address. This *-request* address, if present, will also be used by the *list* channel as the return address for mail submitted to the list.

See Also

deliver(ADM), submit(ADM)

Files

<mmdf-table-directory>/aliases - to find list-*request* addresses

lpadmin

configure the print service

Syntax

/usr/lib/lpadmin -p printer options
/usr/lib/lpadmin -x dest
/usr/lib/lpadmin -d [dest]
/usr/lib/lpadmin -S print-wheel **-A** alert-type [**-W** $integer_1$]
[**-Q** $integer_2$]

Description

lpadmin configures the LP print service to describe printers and devices. It is used to add and change printers, to remove printers from the service, to set or change the system default destination, and to define alerts for print wheels. and to define printers for remote printing services.

Adding or Changing a Printer

The first form of the *lpadmin* command (**lpadmin -p** *printer options*) is used to configure a new printer or to change the configuration of an existing printer. The following options are used and may appear in any order. For ease of discussion, the printer will be referred to as *P* below.

-F *fault-recovery*
Restores the LP print service after a printer fault according to the value of *fault-recovery*:

continue Continues printing on the top of the page where printing stopped. This requires a filter to wait for the fault to clear before automatically continuing.

beginning
Starts printing the request again from the beginning.

wait Disables printing on the printer and waits for the administrator or a user to enable printing again.

During the wait, the administrator or the user who submitted the stopped print request can issue a change request that specifies where printing should resume. If no change request is made before printing is enabled, printing will resume at the top of the page where stopped if the filter allows; otherwise, the request will be printed from the beginning.

This option specifies the recovery to be used for any print request that is stopped because of a printer fault.

-c *class*

Inserts printer *P* into the specified *class*. *class* will be created if it does not already exist.

-D *comment*

Saves *comment* for display whenever a user asks for a full description of the printer *P* [see *lpstat*(C)]. The LP print service does not interpret this comment.

-e *printer*

Copies an existing *printer's* interface program to be the new interface program for printer *P*.

-f allow:*form-list*

-f deny:*form-list*

Allows (**-f allow**) or denies (**-f deny**) the forms in *form-list* to be printed on printer *P*.

For each printer, the LP print service keeps two lists of forms: an li''allow-list'' of forms that can be used with the printer and a ''deny-list'' of forms that shouldn't be used with the printer. With the **-f allow** option, the forms listed are added to the allow-list and removed from the deny-list. With the **-f deny** option, the forms listed are removed from the allow-list and added to the deny-list.

If the allow-list is not empty, the forms in the list can be used with the printer and all others cannot regardless of the content of the deny-list. If the allow-list is empty but the deny-list is not, the forms in the deny-list cannot be used with the printer. All forms can be excluded from a printer by having an empty allow-list and putting the word **any** in the deny-list. All forms can be used on a printer by having an empty deny-list and specifying **any** for the allow-list, provided the printer can handle all the characteristics of the forms.

The LP print service uses this information as a set of guidelines for determining where a form can be mounted. Administrators, however, are not restricted from mounting a form on any printer. If mounting a form on a particular printer is in disagreement with the information in the allow-list or deny-list, the administrator is warned, but the mount is accepted. Nonetheless, if a user attempts to issue a print or change request for a form and printer combination that is in disagreement with the information, the request is accepted only if the form is currently mounted on the printer. If the form is later unmounted before the request can print, the request is canceled, and the user is notified by mail.

If an administrator tries to name a form as acceptable for use on a printer that doesn't have the capabilities needed by the form, the command is rejected.

Note the other use of **-f** below.

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

-i *interface*
Establishes a new interface program for *P*. *interface* is the path name of the new program.

-I *content-type-list*
Assigns *P* to handle print requests with content of a type listed in *content-type-list*.

The type **simple** is recognized as the default content-type of files on the system. Such a data stream contains only printable ASCII characters and the following control characters:

Control Character	Octal Value	Meaning
backspace	10_8	move back to previous column, except at beginning of line
tab	11_8	move to next tab stop
linefeed (newline)	12_8	move to beginning of next line
form feed	14_8	move to beginning of next page
carriage return	15_8	move to beginning of current line

To force the print service to not consider **simple** as a valid type for the printer, give an explicit value (e.g., the printer type) in the *content-type-list*. If you do want **simple** included along with other types, you must include **simple** in the *content-type-list*.

Each printer automatically has its printer type included in the list of content types it will accept.

Except for **simple**, each *content-type* name is freely determined by the administrator. If names given as content types are also printer types, the names are accepted without comment because the LP print service recognizes all printer types as potential content types as well.

-l Indicates that the device associated with *P* is a login terminal. The LP scheduler, *lpsched*, disables all login terminals automatically each time it is started. Before re-enabling *P*, its current *device* should be established using *lpadmin*.

-M -f *form-name* **[-a [-o filebreak]]**

Mounts the form *form-name* on *P*. Print requests to be printed with the pre-printed form *form-name* will be printed on *P*. If more than one printer has the form mounted and the user has specified any (with the **-d** option of the **lp** command) as the printer destination, then each print request will be printed on the one that meets the other needs of the request.

The page length and width and character and line pitches needed by the form are compared with those allowed for the printer by checking the capabilities in the *terminfo*(F) database for the type of printer. If the form requires attributes that are not available with the printer, the administrator is warned, but the mount is accepted. If the form lists a print wheel as mandatory but the print wheel mounted on the printer is different, the administrator is also warned but the mount is accepted.

If the **-a** option is given, an alignment pattern is printed, preceded by the same initialization of the physical printer that precedes a normal print request with one exception: no banner page is printed. Printing is assumed to start at the top of the first page of the form. After the pattern is printed, the administrator can adjust the mounted form in the printer, press return for another alignment pattern (no initialization this time), and continue printing as many alignment patterns as desired. The administrator can quit the printing alignment patterns by typing "q".

If the **-o filebreak** option is given, a formfeed is inserted between each copy of the alignment pattern. By default, the alignment pattern is assumed to correctly fill a form, so no formfeed is added.

A form is unmounted by mounting a new form in its place using the **-f** option. The **-f none** option can be used to specify no form. By default, a new printer has no form mounted.

Note the other use of **-f** above.

-M -S *print-wheel*

Mounts the print wheel *print-wheel* on *P*. Print requests to be printed with *print-wheel* will be printed on *P*. If more than one printer has the *print-wheel* mounted and the user has specified any (with the **-d** option of the **lp** command) as the printer destination, then each print request will be printed on the one that meets the other needs of the request.

If the *print-wheel* is not listed as acceptable for the printer, the administrator is warned, but the mount is accepted. If the printer does not take print wheels, the command is rejected.

A print wheel is unmounted by mounting a new print wheel in its place or by using the **-S none** option.

By default, a new printer has no special print wheel mounted. Until this is changed, a print request that asks for a specific print wheel will not be printed on *P*.

Note the other uses of the **-S** option described below.

-m *model*
Selects a model interface program provided with the LP print service for printer *P*.

-o *printing-option*
Each **-o** option in the list below is the default given to an interface program if the option is not taken from a preprinted form description or is not explicitly given by the user submitting a request [see *lp*(C)]. The only **-o** options that can have defaults defined are listed below:

length=*scaled-decimal-number*
width=*scaled-decimal-number*
cpi=*scaled-decimal-number*
lpi=*scaled-decimal-number*
stty=*stty-option-list*

The term *scaled-decimal-number* refers to a non-negative number used to indicate a unit of size. (The type of unit is shown by a trailing letter attached to the number.) Three types of scaled decimal numbers are discussed for the LP print service: numbers that show sizes in centimeters (marked with a trailing **c**), numbers that show sizes in inches (marked with a trailing **i**), and numbers that show sizes in units appropriate to use (without a trailing letter), i.e., lines, columns, lines per inch, or characters per inch.

The first four default option values should agree with the capabilities of the type of physical printer as defined in the *terminfo*(F) database for the printer type. If they do not, the command is rejected.

The *stty-option-list* is not checked for allowed values but is passed directly to the *stty*(C) program by the standard interface program. Any error messages produced by *stty*(C) when a request is processed (by the standard interface program) are mailed to the user submitting the request.

For each printing option not specified, the defaults for the following attributes are defined in the Terminfo entry for the specified printer type:

length
width
cpi

lpi

The default for **stty** is

> **stty=**9600 cs8 -cstopb -parenb -paroff ixon
> -ixany opost -olcuc -onlcr -ocrnl -onocr
> -onlret -ofill nl0 cr0 tab0 bs0 vt0 ff0

You can set any of the **-o** options to the default values (which vary for different types of printers) by typing them without assigned values as follows:

> **length=**
> **width=**
> **cpi=**
> **lpi=**
> **stty=**

-o nobanner
Allows users to submit a print request that asks that no banner page be printed.

-o banner
Forces a banner page to be printed with every print request, even when a user asks for no banner page. This is the default; you must specify **-o nobanner** if you want to allow users to specify **-o nobanner** with the **lp** command.

-R *machine-list*
Sets up remote machines in *machine-list* to share print services. The LP print service arranges for the advertising and mounting of all necessary resources and for automatic recovery of shared print services when the machine is brought to a state where RFS is run.

The LP Spooler keeps the parts of the print service owned by each machine separate, so that the administrator on one machine can change only the service provided by his or her machine. The LP Spooler provides for no centrally managed print service using RFS.

-r *class*
Removes printer *P* from the specified *class*. If *P* is the last member of the *class*, then the *class* will be removed.

-S *list*
Allows the aliases for character sets or print wheels named in *list* to be used with *P*.

If the printer is a type that takes print wheels, then *list* is a list of print wheel names separated by commas or spaces. These will be the only print wheels considered mountable on the printer. (You can always force a different print wheel to be mounted, however.) Until the option is used to specify a list, no print wheels will be considered mountable on the printer, and print requests that ask for a particular print wheel with this printer will be rejected.

If the printer is a type that has selectable character sets, then *list* is a list of character set name *mappings* or aliases separated by commas or spaces. Each *mapping* is of the form

known-name = *synonym*

known-name is a character set number preceded by **cs**, such as **cs3** for character set three, or a character set name from the Terminfo database **csnm** entry. [See *terminfo*(F) in the *Programmer's Reference Manual.*] If this option is not used to specify a list, only the names already known from the Terminfo database or numbers with a prefix of cs will be acceptable for the printer.

If *list* is the word **none**, the previous print wheel list or character set aliases will be removed.

Note the other uses of the **-S** option.

-T *printer-type*
Assigns the given *printer-type*, a representation of a physical printer of type *printer-type*. *Printer-type* is used to extract data from *terminfo*(F); this data is used to initialize the printer before printing each user's request. Some filters may also use *printer-type* to convert content for the printer. If this option is not used, the default *printer-type* will be **unknown**; no useful information will be extracted from *terminfo*(F), so each user request will be printed without first initializing the printer. Also, this option must be used if the following are to work: **-o cpi=**, **-o lpi=**, **-o width=**, and **-o length=** options of the **lpadmin** and **lp** commands, and the **-S** and **-f** options of the **lpadmin** command.

-u allow:*user-list*

-u deny:*user-list*
Allows (**-u allow**) or denies (**-u deny**) the users in *user-list* access to *P*.

For normal access to each printer, the LP print service keeps two lists of users: an *allow-list* of people allowed to use the printer and a *deny-list* of people denied access to the printer. With the **-u allow** option, the users listed are added to the allow-list and removed from the deny-list. With the **-u deny** option, the users listed are removed from the allow-list and added to the deny-list.

If the allow-list is not empty, the users in the list are allowed access to the printer and all others are denied access, regardless of the content of the deny-list. If the allow-list is empty but the deny-list is not, the users in the deny-list are denied access and all others are allowed. If both lists are empty, all users are allowed access. Access can be denied to all users except the LP print service administrator by putting **any** in the deny-list. To allow everyone access to *P* and effectively empty both lists, put **any** in the allow-list.

-U *dial-info*

Assigns the dialing information *dial-info* to the printer. *dial-info* is used with the *dial*(S) routine to call the printer. Any network connection supported by the Basic Networking Utilities will work. *dial-info* can be either a phone number for a modem connection or a system name for other kinds of connections. Or if **-U direct** is given, no dialing will take place because the name **direct** is reserved for a printer that is directly connected. If a system name is given, it is used to search for connection details from the file **/usr/lib/uucp/Systems** or related files. The Basic Networking Utilities are required to support this option. By default, **-U direct** is assumed.

-v *device*

Associates a new *device* with printer *P*. *device* is the path name of a file that is writable by *lp*. Note that the same *device* can be associated with more than one printer.

-A *alert-type* [**-W** *integer*]

The **-A** option is used to send the alert *alert-type* to the administrator when a printer fault is detected and periodically thereafter until the printer fault is cleared by the administrator. The *alert-types* are

mail

Sends the alert message via mail [see *mail*(C)] to the administrator who issues this command.

write

Writes the message to the terminal on which the administrator is logged in. If the administrator is logged in on several terminals, one is chosen arbitrarily.

quiet

Does not send messages for the current condition. An administrator can use this option to temporarily stop receiving further messages about a known problem. Once the fault has been cleared and printing resumes, messages will again be sent when another fault occurs with the printer.

none

Does not send messages until this command is given again with a different *alert-type*; removes any existing alert definition. No alert will be sent when the printer faults until a different alert-type is used (except quiet).

shell-command

shell-command is run each time the alert needs to be sent. *shell-command* should expect the message as standard input.

If there are blanks embedded in the command, enclose the command in quotes. Note that the **mail** and **write** values for this option are equivalent to the values **mail** *user-name* and **write** *user-name*, respectively, where *user-name* is the current name for the administrator. This will be the login name of the person submitting this command unless he or she has used the **su** command to change to another user ID. If the **su** command has been used to change the user ID, then the *user-name* for the new ID is used.

list

The type of the alert for the printer fault is displayed on the standard output. No change is made to the alert.

The message sent appears as follows:

The print wheel *print-wheel* needs to be mounted
on the printer(s):
printer-list
number-of-requests print requests await this print-wheel.

The printer *printer-name* has stopped printing for the reason given below. Fix the problem and bring the printer back on line. Printing has stopped but will be restarted in a few minutes; issue an enable command if you want to restart sooner.

Unless someone issues a change request

lp -i *request-id* **-P** ...

to change the page-list to print, the current request will be repeated from the beginning.

The reason(s) it stopped (multiple reasons indicate reprinted attempts):

reason

The LP print service can detect printer faults only through an adequate fast filter and only when the standard interface program or a suitable customized interface program is used. Furthermore, the level of recovery after a fault depends on the capabilities of the filter.

If the *printer-name* is **all**, the alerting defined in this command applies to all existing printers.

If the **-W** option is not given or $integer_1$ is zero (which represents **once** and is also the default), only one message will be sent per fault. If this command is not used to arrange fault alerting for a printer, the default procedure is to mail one message to the administrator of the printer per fault.

Restrictions

When creating a new printer, either the **-v** or the **-U** option must be supplied. In addition, only one of the following may be supplied: **-e**, **-i**, or **-m**; if none of these three options are supplied, the model standard is used. The **-h** and **-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).

Removing a Printer Destination

The **-x** *dest* option removes the destination *dest* from the LP print service. If *dest* is a printer and is the only member of a class, then the class will be deleted, too. If *dest* is **all**, all printers and classes are removed. No other options are allowed with **-x**.

Changing the System Default Destination

The **-d** [*dest*] option makes *dest*, an existing destination, the new system default destination. If *dest* is not supplied, then there is no system default destination. No other options are allowed with **-d**.

Setting an Alert for a Print Wheel

-S *print-wheel* **-A** *alert-type* [**-W** *integer*$_1$] [**-Q** *integer*$_2$]

The **-S** *print-wheel* option is used with the **-A** *alert-type* option to send the alert *alert-type* to the administrator as soon as the *print-wheel* needs to be mounted and periodically thereafter. The *alert-types* are

mail Sends the alert message via mail [see *mail*(C)] to the administrator who issues this command.

write Writes the message to the terminal on which the administrator is logged in. If the administrator is logged in on several terminals, one is chosen arbitrarily.

quiet Does not send messages for the current condition. An administrator can use this option to temporarily stop receiving further messages about a known problem. Once the *print-wheel* has been mounted and subsequently unmounted, messages will again be sent when the number of print requests again exceeds the threshold.

none Does not send messages until this command is given again with a different *alert-type* (other than **quiet**).

shell-command
The *shell-command* is run each time the alert needs to be sent. The shell command should expect the message as standard input. If there are blanks embedded in the command, enclose the command in quotes. Note that the **mail** and **write** values for this option are equivalent to the values **mail** *user-name* and **write** *user-name*, respectively, where *user-name* is the current name for the administrator. This will be the login name of the person submitting this command unless he or she has used the **su** command to change to another user ID. If the **su** command has been used to change the user ID, then the *user-name* for the new ID is used.

list The type of the alert for the print wheel is displayed on the standard output. No change is made to the alert.

The printers listed are those that the administrator had earlier specified were candidates for this print wheel. The number ($integer_3$) listed next to each printer is the number of requests eligible for the printer. The number ($integer_4$) shown after the printer list is the total number of requests awaiting the print wheel. It will be less than the sum of the other numbers if some requests can be handled by more than one printer.

If the *print-wheel* is **all**, the alerting defined in this command applies to all print wheels already defined to have an alert.

Only one administrator per print wheel can be alerted. If this command is run by more than one administrator for the same print wheel, the last command run applies.

If the **-W** option is not given or $integer_1$ is zero (which is interpreted as **once** and is also the default), only one message will be sent per need to mount a print wheel. If this command is not used to arrange alerting for a print wheel, no alerts will be sent for the print wheel.

If the **-Q** option is also given, the alert will be made when $integer_2$ print requests that need the print wheel are waiting. If the **-Q** option is not given or $integer_2$ is 1 or the word **any**, a message is sent as soon as anyone submits a print request for the print wheel when it is not mounted.

The **-S** option has a different meaning when used with the **-p** option.

Defining Remote Printers for Remote Printing Services

The fourth form of the *lpadmin* command is used to define the remote printer, *printer-name*$_2$, and its machine, *machine-name*, that will handle remote print requests from the local machine. The remote printer will be be referred to as *printer-name*$_1$ on the local machine.

Files

/usr/spool/lp/*

See Also

accept(ADM), enable(C), lp(C), lpstat(C), stty(C), lpsched(ADM), terminfo(F)

Authorization

Permission to use this utility is assigned with the **lp** authorization, which is usually reserved for system administrators.

lpfilter

administer filters used with the print service

Syntax

/usr/lib/lpfilter **-f** filter-name **-F** path-name
/usr/lib/lpfilter **-f** filter-name **-**
/usr/lib/lpfilter **-f** filter-name **-i**
/usr/lib/lpfilter **-f** filter-name **-x**
/usr/lib/lpfilter **-f** filter-name **-l**

Description

The *lpfilter* command is used to add, change, delete, and list filters used with the LP print service. These filters are used to convert the content type of a file to a content type acceptable to a given printer. One of the following options must be used with the *lpfilter* command: **-F** *path-name* (or **-** for standard input) to add or change a filter, **-i** to reset an original LP print service filter to its factory setting, **-x** to delete a filter, or **-l** to list a filter description.

The argument **all** can be used instead of a *filter-name* with any of these options. When **all** is specified with the **-F** or **-** option, the requested change is made to all filters. Using **all** with the **-i** option has the effect of restoring to their original settings all filters for which predefined settings were initially available. Using the **all** argument with the **-l** option produces a list of all filters, and using it with the **-x** option results in all filters being deleted.

Adding or Changing a Filter

The filter named in the **-f** option and described in the input is added to the filter table. If the filter already exists, its description is changed to reflect the new information in the input. Once added, a filter is available for use.

The filter description is taken from the *path-name* if the **-F** option is given or from the standard input if the **-** option is given. One of the two must be given to define or change a filter. If the filter named is one originally delivered with the LP print service, the **-i** option will restore the original filter description.

Filters are used to convert the content of a request into a data stream acceptable to a printer. For a given print request, the LP print service will know the following:

- the type of content in the request
- the name of the printer
- the type of the printer
- the types of content acceptable to the printer
- the modes of printing asked for by the originator of the request

It will use this information to find a filter that will convert the content into a type acceptable to the printer.

Below is a list of items that provide input to this command and descriptions of each item. All lists are separated by commas or spaces.

Input types: *content-type-list*
Output types: *content-type-list*
Printer types: *printer-type-list*
Printers: *printer-list*
Filter type: *filter-type*
Command: *shell-command*
Options: *template-list*

Input types
This gives the types of content that can be accepted by the filter.

Output types
This gives the types of content that the filter can produce from any of the input content types.

Printer types
This gives the type of printers for which the filter can be used. The LP print service will restrict the use of the filter to these types of printers.

Printers
This gives the names of the printers for which the filter can be used. The LP print service will restrict the use of the filter to just the printers named.

Filter type
This marks the filter as a "slow" filter or a "fast" filter. Slow filters are generally those that take a long time to convert their input. They are run unconnected to a printer to keep the printers from being tied up while the filter is running. Fast filters are generally those that convert their input quickly or those that must be connected to the printer when run. These will be given to the interface program to run connected to the physical printer.

Command

This specifies the program to run to invoke the filter. The program name as well as fixed options are included in the *shell-command*; additional options are constructed, based on the characteristics of each print request and on the **Options** field.

The command must accept a data stream as standard input and produce the converted data stream on its standard output. This allows filter pipelines to be constructed to convert data not handled by a single filter.

Options

This is a list of templates separated by commas used by the LP print service to construct options to the filter from the characteristics of each print request listed in the table later. In general, each template is of the following form:

keyword pattern = replacement

The *keyword* names the characteristic that the template attempts to map into a filter-specific option; each valid *keyword* is listed in the table below. A *pattern* is either a literal pattern of one of the forms listed in the table or a single asterisk, *; if the *pattern* matches the value of the characteristic, the template fits and is used to generate a filter-specific option. A *pattern* of * matches any value. The *replacement* is a string used as a filter-specific option with an embedded asterisk, *, replaced with the value of the characteristic.

lp Option	Characteristic	*keyword*	Possible
-T	Content type (input)	**INPUT**	*content-type*
N/A	Content type (output)	**OUTPUT**	*content-type*
N/A	Printer type	**TERM**	*printer-type*
-f, **-o** cpi=	Character pitch	**CPI**	*integer*
-f, **-o** lpi=	Line pitch	**LPI**	*integer*
-f, **-o** length=	Page length	**LENGTH**	*integer*
-f, **-o** width=	Page width	**WIDTH**	*integer*
-P	Pages to print	**PAGES**	*page-list*
-S	Character set/ print wheel	**CHARSET**	*character-set-name/ print-wheel-name*
-f	Form name	**FORM**	*form-name*
-y	Modes	**MODES**	*mode*
-n	Number of copies	**COPIES**	*integer*

For example, the template

MODES landscape = -l

would show that if a print request includes the **-y landscape** option, the filter should be given the option **-l**. As another example, the template

TERM * = -T *

would show that the filter should be given the option **-T** *printer-type* for whichever *printer-type* is associated with a print request using the filter.

When an existing filter is changed with this command, items that are not specified in the new information are left as they were. When a new filter is added with this command, unspecified items are given default values.

Note that a filter name and a command must be given. A filter with no input type value is assumed to work with any input type; this is also true for the output type, printer type, and printer values.

Deleting a Filter

The **-x** option is used to delete the filter specified in *filter-name* from the LP filter table.

Listing a Filter Description

The **-l** option is used to list the description of the filter named in *filter-name*. If the command is successful, the following message is sent to standard output:

Input types: *content-type-list*
Output types: *content-type-list*
Printer types: *printer-type-list*
Printers: *printer-list*
Filter type: *filter-type*
Command: *shell-command*
Options: *template-list*

If the command fails, an error message is sent to standard error.

See Also

lpadmin(ADM), lp(C)

Authorization

The use of this utility is governed by assignment of the **lp** authorization, which is usually reserved for system administrators.

lpforms

administer forms used with the print service

Syntax

/usr/lib/lpforms **-f** form-name *option*

/usr/lib/lpforms **-f** form-name **-A** alert-type [**-Q** *integer* $_1$] [**-W** *integer* $_2$]

/usr/lib/lpforms **-f** *form-name* **-A list**

/usr/lib/lpforms **-f** *form-name* **-A quiet**

/usr/lib/lpforms **-f** *form-name* **-A none**

Description

The *lpforms* command is used to administer the use of preprinted forms, such as company letterhead paper, with the LP print service. The first variation of the *lpforms* command allows the administrator to add, change, and delete forms, to list the attributes of an existing form, and to allow and deny users access to particular forms. The second variation of *lpforms* is used to establish the method by which the administrator is alerted that a form must be mounted on a printer. The third variation is used to list the current alerting methods assigned to forms. The form is specified by the *form-name* given with the *lpforms* command. Users may request this form by *form-name* [see *lp*(C)]. The fourth variation of *lpforms* is to terminate an active alert. The fifth form is used to remove an alert.

With the first variation of the *lpforms* command, one of the following options must be used:

-F *path-name* to add or change a form as specified by the information in *path-name*

- To add or change a form, and supply information from standard input

-x to delete a form

This option must be used separately; it cannot be used with any other option.

-l to list the attributes of a form

This option must be used separately; it cannot be used with any other option.

-u allow:*user-list*
to allow users to request a form

This option can be used with the **-F** or **-** option.

-u deny:*user-list*
to deny users access to a form

This option can be used with the **-F** or **-** option.

Each option is explained below.

Adding or Changing a Form

The **-F** *path-name* option is used to add a new form to the LP print service, or to change the attributes of an existing form. The form description is taken from *path-name* if the **-F** option is given, or the standard input if the **-** option is given. One of the two options must be given to define or change a form. *path-name* is the path name of a file that contains all or any subset of the following information about the form.

Page length: *scaled-decimal-number*$_1$
Page width: *scaled-decimal-number*$_2$
Number of pages: *integer*
Line pitch: *scaled-decimal-number*$_3$
Character pitch: *scaled-decimal-number*$_4$
Character set choice: *character-set/print-wheel*,**[mandatory]**
Ribbon color: *ribbon-color*
Comment:
comment
Alignment pattern: [*content-type*]
content

Except for the last two lines, the above lines can appear in any order. The **Comment:** and *comment* items must appear in consecutive order but can appear before the other items, and the **Alignment pattern:** and the *content* items must appear in consecutive order at the end of the file. Also, the *comment* item cannot contain a line that begins with any of the key phrases above, unless the key phrase is preceded with a "**>**" sign. Any leading > sign found in the *comment* will be removed when the comment is displayed. Case distinctions in the key phrases are ignored.

Upon issuing this command, the form named in *form-name* is added to the list of forms. If the form already exists, its description is changed to reflect the new information in the input. Once added, a form is available for use in a print request, except where access to the form has been restricted, as described under the **-u allow:** option. A form may also be allowed to be used on certain printers only.

A description of each form attribute is below:

Page length and **Page Width**

Before printing the content of a print request needing this form, the generic interface program provided with the LP print service will initialize the physical printer to handle pages *scaled-decimal-number*$_1$ long, and *scaled-decimal-number*$_2$ wide using the printer type as a key into the *terminfo*(F) database. A *scaled-decimal-number* is an optionally scaled decimal number that gives a size in lines, columns, inches, or centimeters, as appropriate. The scale is indicated by appending the letter 'i', for inches, or the letter 'c', for entimeters. For length or width settings, an unscaled number indicates lines or columns; for line pitch or character pitch settings, an unscaled number indicates lines per inch or characters per inch (the same as a number scaled with 'i'). For example, **length=66** indicates a page length of 66 lines, **length=11i** indicates a page length of 11 inches, and **length=27.94c** indicates a page length of 27.94 centimeters.

The page length and page width will also be passed, if possible, to each filter used in a request needing this form.

Number of pages

Each time the alignment pattern is printed, the LP print service will attempt to truncate the *content* to a single form by, if possible, passing to each filter the page subset of 1-*integer*.

Line pitch and **Character pitch**

Before printing the content of a print request needing this form, the interface programs provided with the LP print service will initialize the physical printer to handle these pitches, using the printer type as a key into the *terminfo*(F) database. Also, the pitches will be passed, if possible, to each filter used in a request needing this form. *Scaled-decimal-number*$_3$ is in lines per centimeter if a 'c' is appended, and lines per inch otherwise; similarly, *scaled-decimal-number*$_4$ is in columns per centimeter if a 'c' is appended, and columns per inch otherwise. The character pitch can also be given as **elite** (12 characters per inch), **pica** (10 characters per inch), or **compressed** (as many characters per inch as possible).

Character set choice

When the LP print service alerts an administrator to mount this form, it will also mention that the print wheel *print-wheel* should be used on those printers that take print wheels. If printing with this form is to be done on a printer that has selectable or loadable character sets instead of print wheels, the interface programs provided with the LP print service will automatically select or load the correct character set. If **mandatory** is appended, a user is not allowed to select a different character set for use with the form; otherwise, the character set or print wheel named is a suggestion and a default only.

Ribbon color

When the LP print service alerts an administrator to mount this form, it will also mention that the color of the ribbon should be *ribbon-color*.

Comment

The LP print service will display the *comment* unaltered when a user asks about this form [see *lpstat*(C)].

Alignment pattern

When mounting this form an administrator can ask that the *content* be repeatedly printed, as an aid in correctly positioning the pre-printed form. The optional *content-type* defines the type of printer for which *content* had been generated. If *content-type* is not given, **simple** is assumed. Note that the *content* is stored as given, and will be readable only by the user **lp**.

When an existing form is changed with this command, items missing in the new information are left as they were. When a new form is added with this command, missing items will get the following defaults:

Page Length: **66**
Page Width: **80**
Number of Pages: **1**
Line Pitch: **6**
Character Pitch: **10**
Character Set Choice: **any**
Ribbon Color: **any**
Comment: (no default)
Alignment Pattern: (no default)

Deleting a Form

The **-x** option is used to delete the form specified in *form-name* from the LP print service.

Listing Form Attributes

The **-l** option is used to list the attributes of the existing form specified by *form-name*. The attributes listed are those described under "Adding and Changing a Form," above. Because of the potentially sensitive nature of the alignment pattern, only the administrator can examine the form with this command. Other people can use the *lpstat*(C) command to examine the non-sensitive part of the form description.

Allowing and Denying Access to a Form

The LP print service keeps two lists of users for each form, an allow-list of people allowed to use the form, and a deny-list of people denied access to the form. With the **-u allow:** option, the users listed are added to the allow-list and removed from the deny-list. With the **-u deny:** option, the users listed are removed from the allow-list and added to the deny-list.

If the allow-list is not empty, the users in the list are allowed access to the form and all others are denied access, regardless of the content of the deny-list. If the allow-list is empty, but the deny-list is not, the users in the deny-list are denied access and all others are allowed. If both lists are empty, all users are allowed access. Access can be denied to all users, except the LP print service administrator, by putting **any** in the deny-list. To effectively empty both lists, allowing access for everyone, put **any** in the allow-list.

Alerting to Mount Forms

The second variation of the *lpforms* command is used to arrange for the alerting to mount forms on a printer.

When *integer* $_1$ print requests needing the preprinted form *form-name* become queued up because no printer satisfying all the needs of the requests has the form mounted, and for as long as this condition remains, an alert is sent to the administrator every *integer* $_2$ minutes until the form is mounted on a qualifying printer. If the *form-name* is **all**, the alerting defined in this command applies to all existing forms. No alerting is done for a backlog of print requests needing a form if the administrator does not use this option.

The method for sending the alert depends on the value of the **-A** option.

write
The message is sent via *write*(C) to the terminal on which the administrator is logged in when the alert arises. If the administrator is logged in on several terminals, one is chosen arbitrarily.

mail
The message is sent via mail to the administrator who issues this command.

The message sent appears as follows:

The form *form-name needs to be mounted on the printer(s).*
printer-list (*integer* $_3$ *requests*)
integer $_4$ print request awaits this form.
Use the *ribbon-color ribbon.*
Use the print-wheel print wheel, if appropriate.

The printers listed are those that the administrator had earlier specified were candidates for this form. The number (*integer* $_3$) listed next to each printer is the number of requests eligible for the printer. The number (*integer* $_4$) shown after the printer list is the total number of requests awaiting the form. It will be less than the sum of the other numbers if some requests can be handled by more than one printer. The *ribbon-color* and *print-wheel* are those given in the form description. The last line in the message is given even if none of the printers listed use print wheels, because the administrator may choose to mount the form on a printer that does use a print wheel.

Where any color ribbon or any print wheel can be used, the statements above will read:

Use any ribbon.
Use any print-wheel.

shell-command
The *shell-command* is run each time the alert needs to be sent. The shell command should expect the message as standard input. Note that the **mail** and **write** values for the **-A** command are equivalent to the values **mail** *user-name* and **write** *user-name*, respectively, where *user-name* is the current name for the administrator. This will be the login name of the person submitting this command *unless* he or she has used the *su* command to change to another user ID. If the *su* command has been used to change the user ID, then the *user-name* for the new ID is used.

If the **-Q** option is not given or *integer* $_1$ is one or the word **any** (which is the default), a message is sent as soon as anyone submits a print request for the form when it is not mounted.

If the **-W** option is not given or *integer* $_2$ is zero or the word **once** (which is the default), only one message is sent when the queue size exceeds *integer* $_1$.

Listing the Current Alert

The third variation of *lpforms* is used to list the type of the alert for the specified form. No change is made to the alert. If *form-name* is recognized by the LP print service, one of the following lines is sent to the standard output, depending on the type of alert for the form.

When *integer are queued: alert with shell-command every integer minutes*

When *integer are queued: write to user-name every integer minutes*

When *integer are queued: mail to user-name every integer minutes*

No alert

The phrase every *integer* minutes *is replaced with once* if $integer_2$ (the **-W** $integer_2$) is 0.

Terminating an Active Alert

The **quiet** option is used to stop messages for the current condition. An administrator can use this option to temporarily stop receiving further messages about a known problem. Once the form has been mounted and then unmounted, messages will again be sent when the queue size reaches $integer_1$ pending requests.

Removing an Alert Definition

No messages will be sent until the **none** option is given again with a different *alert-type*. This can be used to permanently stop further messages from being sent.

See Also

lp(C), lpadmin(ADM), terminfo(F)

Authorization

The use of this utility is governed by assignment of the **lp** authorization, which is usually reserved for system administrators.

lpsched, lpshut, lpmove

start/stop the print service and move requests

Syntax

/usr/lib/lpsched
/usr/lib/lpsched -q *integer*
/usr/lib/lpsched -a *integer*
/usr/lib/lpsched -p *integer*
/usr/lib/lpsched -s *integer*
/usr/lib/lpshut
/usr/lib/lpmove requests dest
/usr/lib/lpmove dest1 dest2

Description

lpsched starts the LP print service; this can be done only by **root** or **lp**.

lpshut shuts down the print service. All printers that are printing at the time *lpshut* is invoked will stop printing. When *lpsched* is started again, requests that were printing at the time a printer was shut down will be reprinted from the beginning.

lpmove moves requests that were queued by **lp** between LP destinations. The first form of the command moves the named *requests* to the LP destination *dest*. *Requests* are *request-ids* as returned by **lp**. The second form moves all requests for destination *dest1* to destination *dest2*; **lp** will then reject any new requests for *dest1*.

Note that when moving requests, *lpmove* never checks the acceptance status (see *accept*(ADM)) of the new destination. Also, the request ID of the moved request is not changed so that users can still find their requests. The *lpmove* command will not move requests that have options (content type, form required, and so on) that cannot be handled by the new destination.

-q *integer*
Specify the number of request structures you want to allocate.

-a *integer*
Specify the number of alert structures you want to allocate. By default, forty empty alert structures are allocated in addition to one for each printer or form on the system. Structures will always be allocated for existing printers and forms. You can choose, however, to have more or fewer than the forty extra, by using the **-a** option. For example, if you want only as many alert structures as you have printers and forms on your system, enter

the following command: **lpsched -a 0**.

-p *integer*
Specify the number of print status structures you want to allocate. By default, twenty-five empty printer status structures are allocated in addition to one for each printer on the system. Structures will always be allocated for existing printers. You can choose, however, to have more or fewer than the forty extra, by using the **-p** option.

-s *integer*
Specify the number of slow filters per printer that can be run simultaneously.

Notes

By default, the directory **/usr/spool/lp** is used to hold all the files used by the LP print service. This can be changed by setting the **SPOOLDIR** environment variable to another directory before running *lpsched*. If you do this, you should populate the directory with the same files and directories found under **/usr/spool/lp**; the LP print service will not automatically create them. Also, the **SPOOLDIR** variable must then be set before any of the other LP print service commands are run.

Files

/usr/spool/lp/*

See Also

enable(C), lp(C), lpstat(C), accept(ADM), lpadmin(ADM)

Authorization

The behavior of this utility is affected by assignment of the **lp** authorization, which is usually reserved for system administrators.

lpsh

menu driven lp print service administration utility

Syntax

/usr/lib/sysadm/lpsh

Description

lpsh is the screen interface invoked by the *sysadmsh*(ADM) Printers selection to administer the print service. The interface performs all of the required lp print service functions that require system administrator authorization, **lp**.

The program allows the administrator to perform any of the following tasks:

- configure the LP print service to describe printers and devices.
- administer filters to be used with the LP print service.
- administer forms to be used with the LP print service.
- start the LP print service.
- shut down the LP print servce.
- move print requests between printer destination.
- cancel print requests.
- allow destinations to accept or reject print requests.
- set the printing queue priorities that can be assigned to jobs submitted by users of the LP print service.
- enable or disable printers.

See Also

auditsh(ADM), authsh(ADM), backupsh(ADM), atcronsh(ADM), sysadmsh(ADM), lp(C), lpadmin(ADM), lpfilter(ADM), lpforms(ADM), lpsched(ADM), lpusers(ADM), accept(ADM), enable(C)

Notes

Invoking the *lpsh* directly is not recommended; use the *sysadmsh* Printers selection.

Value Added

lpsh is an extension of AT&T System V provided in Altos UNIX System V.

lpusers

set printing queue priorities

Syntax

/usr/lib/lpusers -d priority-level
/usr/lib/lpusers -q priority-level **-u** user-list
/usr/lib/lpusers -u user-list
/usr/lib/lpusers -q priority-level
/usr/lib/lpusers -l

Description

The *lpusers* command is used to set limits to the queue priority level that can be assigned to jobs submitted by users of the LP print service.

The first form of the command (with **-d**) sets the system-wide priority default to *priority-level*, where *priority-level* is a value of 0 to 39, with 0 being the highest priority. If a user does not specify a priority level with a print request [see *lp*(C)], the default priority is used. Initially, the default priority level is 20.

The second form of the command (with **-q** and **-u**) sets the default highest *priority-level* (0-39) that the users in *user-list* can request when submitting a print request. Users that have been given a limit cannot submit a print request with a higher priority level than the one assigned, nor can they change a request already submitted to have a higher priority. Any print requests with priority levels higher than allowed will be given the highest priority allowed.

The third form of the command (with **-u**) removes the users from any explicit priority level and returns them to the default priority level.

The fourth form of the command (with **-q**) sets the default highest priority level for all users not explicitly covered by the use of the second form of this command.

The last form of the command (with **-l**) lists the default priority level and the priority limits assigned to users.

See Also

lp(C)

majorsinuse

displays the list of major device numbers currently specified in the mdevice file

Syntax

/etc/conf/cf.d/majorsinuse

Description

This script searches the **mdevice** file and displays a list of the major device numbers already in use.

When installing a device driver with the Link Kit, you can use *majorsinuse* to find an available major device number for the driver. When you invoke the *configure* program to modify the system configuration files with the new driver information, use the **-m** option to indicate the major device number of the driver.

The **-j** option to *configure* performs a function similar to that of *majorsinuse*. If you give the **-j** option with the NEXTMAJOR keyword, *configure* tells you the next available major device number.

Files

/etc/conf/cf.d/mdevice

See Also

configure(ADM), mdevice(F), ''Adding Device Drivers with the Link Kit'' in the *System Administrator's Guide*

Value Added

majorsinuse is an extension of AT&T System V provided in Altos UNIX System V.

makekey

generates an encryption key

Syntax

/usr/lib/makekey

Description

makekey improves the usefulness of encryption schemes by increasing the amount of time required to search the key space. It reads 10 bytes from its standard input, and writes 13 bytes on its standard output. The output depends on the input in a way that is intended to be difficult to compute (i.e., to require a substantial fraction of a second).

The first 8 input bytes (the *input key*) can be arbitrary ASCII characters. The last 2 input bytes (the *salt*) are best chosen from the set of digits, dot (.), slash (/), and uppercase and lowercase letters. The *salt* characters are repeated as the first 2 characters of the output. The remaining 11 output characters are chosen from the same set as the *salt* and constitute the *output key*.

The transformation performed is essentially the following: the *salt* is used to select one of 4,096 cryptographic machines all based on the National Bureau of Standards DES algorithm, but broken in 4,096 different ways. Using the *input key* as the key, a constant string is fed into the machine and recirculated. The 64 bits that come out are distributed into the 66 *output key* bits in the result.

See Also

passwd(F)

mkdev

calls scripts to add peripheral devices

Syntax

/etc/mkdev dos
/etc/mkdev fd
/etc/mkdev fs [device file]
/etc/mkdev hd [[disk] [adapt_#] [LUN] [adapt_type]]
/etc/mkdev mouse
/etc/mkdev serial
/etc/mkdev shl
/etc/mkdev streams
/etc/mkdev tape [[contrlID] [adapt_#] [LUN] [adapt_type]]
/etc/mkdev graphics

Description

mkdev creates the device file(s) associated with a peripheral device. Based on the argument supplied, the *mkdev* command calls a script found in the directory **/usr/lib/mkdev**. If no arguments are listed, *mkdev* prints a usage message. The following paragraphs describe each of the optional arguments that can be used with *mkdev*:

/etc/mkdev dos initializes necessary devices and configures the system to support mounted DOS filesystems.

/etc/mkdev hd creates device files for use with a peripheral hard disk. The device files for an internal hard disk already exist. */etc/mkdev hd* invokes the following utilities: *dparam*(ADM), *fdisk*(ADM), and *divvy*(ADM). SCSI disks require four pieces of information: the ID of the disk controller, the host adapter number, the LUN (logical unit number, which always should be zero), and the host adapter type. Thus, you can install SCSI disks with a *mkdev hd* command containing arguments within the following ranges:

mkdev hd [0-6] [0-3] 0 [ti | hpfp | ad]

You must invoke mkdev hd twice to install a SCSI disk. The first time, the kernel will be reconfigured to support the new disk. The second time, the disk will be initialized. Use the same *mkdev hd* arguments both times. However, if the kernel has already been configured to support the new disk, *mkdev hd* need only be invoked once to initialize the disk.

/etc/mkdev serial creates device files for use with serial cards. The device files for the first and second ports already exist. Additional device files must be created for the ports added when expansion cards are added to the system.

/etc/mkdev streams configures the kernel for streams support.

/etc/mkdev fs performs the system maintenance tasks required to add a new filesystem to the system once the device is created (*mknod*(C)) and the filesystem is made (*mkfs*(ADM)). It creates the */file* and */file*/**lost+found** directories, reserves slots in the **lost+found** directory, (if either already exist, they are used unmodified) and modifies **/etc/checklist**, **/etc/default/filesys** and **/etc/default** to check (*fsck*(ADM)) and mount (*mount*(ADM), *mnt*(C), *rc*(C)) the filesystem as appropriate. It is usually used in conjunction with *mkdev hd* when adding a second hard disk to the system or with *mkdev fd* when creating a mountable filesystem on a floppy, but can be used on any additional filesystem (for example, on a large internal hard disk).

/etc/mkdev fd creates bootable, root and filesystem floppy disks.

Several floppies can be created during a single *mkdev fd* session, but *mkdev* does not display a prompt to remove the first floppy and insert the next one. Insert the next floppy when *mkdev* prompts "Would you like to format the floppy first? (y/n)."

/etc/mkdev tape configures the tape driver in preparation for linking a new kernel that includes tape support. It adds a standard quarter-inch cartridge tape driver and/or a mini-cartridge tape driver.

The current driver configurations can be displayed, and changed if necessary. A zero in any of the fields means the driver automatically detects the type of tape device installed and uses the built-in values for that device. If the autoconfiguration values are not correct for your drive, refer to your hardware manual for the correct values, configure the driver and relink the new kernel. *mkdev tape* can also be used to remove a tape driver from the existing kernel.

SCSI tapes, such as Exabyte or 9-track, can also be installed using */etc/mkdev tape*. Just as with SCSI disks, the adapter type, adapter number, and the controller ID must be specified (the LUN should always be zero).

/etc/mkdev shl initializes necessary devices and configures kernel parameters associated with the number of shell layers sessions available on the system.

/etc/mkdev mouse initializes necessary devices and configures the system to use any supported mouse.

Use */etc/mkdev graphics* to up the graphics device (display adapter) for the console and any of its virtual terminals. When you invoke *mkdev graphics*, you are placed in a menu-driven interface similar to the interface used for *sysadmsh*(ADM). With these menus you can control three basic aspects of the graphics adapter: Adapter Type, Adapter Mode (resolution), and affected terminals (Devices).

The Adapter Type field lists all display devices that are supported by the operating system. Select the device currently installed in your system (e.g., Orchid Designer VGA). The Adapter Mode associates the adapter type with a resolution (via the **grafinfo** file). The Devices field lets you select which virtual terminals will be controlled by this adapter type and resolution combination. See *multiscreen*(M) for more information on virtual console terminals.

A common use of *mkdev graphics* is to reconfigure your adapter to run at a higher resolution, for example when running graphics-intensive applications software (such as Open Desktop). However, note that you may set your graphics adapter to run only at those resolutions that are supported by the **grafinfo** file, regardless of any additional resolutions the adapter hardware supports.

Once the driver is configured, you are prompted for re-linking the kernel. The appropriate devices in **/dev** are created.

The various *init* scripts prompt for the information necessary to create the devices.

Files

/usr/lib/mkdev/*
/usr/lib/grafinfo/*

See Also

badtrk(ADM), divvy(ADM), dparam(ADM), fd(HW), fdisk(ADM), filesys(F), format(C), hd(HW), lp(HW), mkfs(ADM), mknod(C), mount(ADM), serial(HW), usemouse(C), tape(HW)

The *System Administrator's Guide* has chapters devoted to the installation of most peripheral devices.

Value Added

mkdev is an extension of AT&T System V provided in Altos UNIX System V.

mkfs

constructs a filesystem

Syntax

/etc/mkfs [**-y** | **-n**] [**-f** fstype] special blocks[: inodes] [gap inblocks]
/etc/mkfs special proto [gap inblocks]

XENIX filesystem options

[**-s** blocks [: inodes]]

UNIX filesystem options

[**-b** blocksize]

AFS filesystem options

[**-c** clustersize]

Description

mkfs constructs a file system by writing on the special file *special* , according to the directions found in the remainder of the command line. *mkfs* is actually a front-end that invokes the appropriate version of *mkfs* according to the filesystem type. The **-f** option specifies the filesystem type, which can be one of the following:

AFS (Acer Fast Filesystem)
S51K (UNIX)
XENIX
DOS

The AFS is the default filesystem type.

Standard Options

If it appears that the special file contains a file system, operator confirmation is requested before overwriting the data. The **-y** "yes" option overrides this, and writes over any existing data without question. The **-n** option causes *mkfs* to terminate without question if the target contains an existing file system. The check used is to read block one from the target device (block one is the super-block) and see whether the bytes are the same. If they are not, this is taken to be meaningful data and confirmation is requested.

If the second argument to *mkfs* is a string of digits, the size of the file system is the value of *blocks* interpreted as a decimal number. This is the number of *physical* (512-byte) disk blocks the file system will occupy. If the number of inodes is not given, the default is approximately the number of *logical* blocks divided by 4. *mkfs* builds a file system with a single empty directory on it. The boot program block (block zero) is left uninitialized.

If the second argument is the name of a file that can be opened, *mkfs* assumes it to be a prototype file *proto*, and will take its directions from that file. 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              $
11      $
```

Line 1 in the example is the name of a file to be copied onto block zero as the bootstrap program.

Line 2 specifies the number of *physical* (512-byte) blocks the file system is to occupy and the number of inodes in the file system.

Lines 3-9 tell *mkfs* about files and directories to be included in this file system.

Line 3 specifies the root directory.

Lines 4-6 and 8-9 specify 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 10 and 11 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 **-bcd** to specify regular, block special, character special and directory files, respectively. The second character of the mode is either **u** or **-** to specify set-user-id mode or not. The third is **g** or **-** 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 from which 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 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 **$**.

The *gap inblocks* argument in both forms of the command specifies the rotational gap and the number of blocks/cylinder.

XENIX filesystem options

The **-s** option is a command-line override of the size and number of inodes in the *proto* file.

UNIX filesystem options

The **-b** *blocksize* option specifies the logical block size for the file system. The logical block size is the number of bytes read or written by the operating system in a single I/O operation. Valid values for *blocksize* are 512, 1024, and 2048. The default is 1024. A block size of 2048 may be chosen only if the 2K file system package is installed. If the **-b** option is used, it must appear last on the command line.

AFS filesystem options

The **-C***clustersize* option specifies the cluster size for the filesystem. This only applies to AFS; if this is included on the command line, the filesystem created will be AFS regardless of the other options used.

Files

/etc/vtoc/*

See Also

chmod(C), dir(F), filesystem(F)

Notes

With a prototype file, it is not possible to copy in a file larger than 64K bytes, nor is there a way to specify links. The maximum number of inodes configurable is 65500.

The directory **/etc/fscmd.d/***TYPE* contains programs for each file system type; *mkfs* invokes the appropriate binary.

mmdf

routes mail locally and over any supported network

Description

The operating system uses MMDF (the Multi-channel Memorandum Distribution Facility) to route mail locally and over Micnet, UUCP, or other networks that provide MMDF support. The *custom* utility installs MMDF and configures a basic system for sending mail on a local machine.

MMDF is a very versatile and configurable mail routing system. MMDF configuration begins with the **/usr/mmdf/mmdftailor** file, which defines the machine and domain names, the various tables (alias, domain, channel), and other configuration information. To change the configuration of MMDF on your system, you can log in as *mmdf* and edit the configuration files. Whenever you change MMDF alias or routing information in any way, you must rebuild the hashed database.

Files

/usr/mmdf/mmdftailor
/usr/mmdf/table/alias.list
/usr/mmdf/table/alias.user
/usr/mmdf/table/*.chn
/usr/mmdf/table/*.dom
/usr/spool/mail/*
/usr/spool/mmdf/...

See Also

dbmbuild(ADM), mmdfalias(ADM), mnlist(ADM), uulist(ADM), tables(F), mmdftailor(F), "Setting Up Electronic Mail" in the *System Administrator's Guide*.

Value Added

mmdf is an extension of AT&T System V provided in Altos UNIX System V.

mmdfalias

converts XENIX-style aliases file to MMDF format

Syntax

/usr/mmdf/table/tools/mmdfalias

Description

mmdfalias is a conversion utility to produce MMDF-compatible alias files from the XENIX-format aliases file. *mmdfalias* also splits the converted contents of **/usr/lib/mail/aliases** into two MMDF files containing list-type aliases and aliases that map users to machines.

After installing MMDF with *custom*, restore **/usr/lib/mail/aliases** from backup tape. Place the following line in the file to indicate where the list aliases end and the mapping aliases begin.

```
# user-to-machine mapping
```

Log in as *mmdf* and run the */usr/mmdf/table/tools/mmdfalias* conversion script from the **/usr/mmdf/table** directory. You now have two MMDF files, **alias.list** and **alias.user**, in the current directory.

After creating these files in **/usr/mmdf/table**, you must rebuild the MMDF hashed database. While logged in as *mmdf*, run *dbmbuild* from **/usr/mmdf/table**.

Files

/usr/lib/mail/aliases
/usr/mmdf/table/alias.list
/usr/mmdf/table/alias.user

See Also

dbmbuild(ADM), tables(F), "Setting Up Electronic Mail" in the *System Administrator's Guide*

Value Added

mmdfalias is an extension of AT&T System V provided in Altos UNIX System V.

mnlist

converts a XENIX-style Micnet routing file to MMDF format

Syntax

/usr/mmdf/table/tools/mnlist

Description

mnlist is a conversion utility to produce MMDF-compatible Micnet routing files from the XENIX-format Micnet routing file.

After installing MMDF with *custom*, restore **/usr/lib/mail/top** from backup media. Log in as *mmdf* and run the conversion script */usr/mmdf/table/tools/mnlist* from the **/usr/mmdf/table** directory. You now have a Micnet channel file, **micnet.chn**, in the current directory.

After creating these files in **/usr/mmdf/table**, you must rebuild the MMDF hashed database. While logged in as *mmdf*, run *dbmbuild* from **/usr/mmdf/table.**

Files

/usr/lib/mail/top
/usr/mmdf/table/micnet.chn

See Also

dbmbuild(ADM), tables(F), ''Setting Up Electronic Mail'' in the *System Administrator's Guide*

Value Added

mnlist is an extension of AT&T System V provided in Altos UNIX System V.

mount

mounts and unmounts a file structure

Syntax

/etc/mount [**-r**] [**-f** fstyp] special directory

/etc/umount special-device

Description

mount announces to the system that a removable file structure is present on *special-device.* The file structure is mounted on *directory.* The *directory* must already exist; it becomes the name of the root of the newly mounted file structure. *directory* should be empty. If *directory* contains files, they will appear to have been removed while the *special-device* is mounted and reappear when the *special-device* is unmounted.

The *mount* and *umount* commands maintain a table of mounted devices. If *mount* is invoked without any arguments, it displays the name of each mounted device, and the directory on which it is mounted, whether the file structure is read-only, and the date it was mounted.

The **-f** *fstyp* option indicates that *fstyp* is the file system type to be mounted. If this argument is omitted, it defaults to the **root** *fstyp.*

The optional **-r** argument indicates that the file is to be mounted read-only. Physically write-protected file structures, such as floppy disks with write-protect tabs, must be mounted in this way or errors occur when access times are updated, whether or not any explicit write is attempted.

umount removes the removable file structure on device *special-device.* Any pending I/O for the file system is completed and the file structure is marked as clean.

Files

/etc/mnttab	Mount table
/etc/default/filesys	Filesystem data

See Also

umount(ADM), mnt(C), mount(S), mnttab(F), default(F), setmnt(ADM)

Diagnostics

mount issues a warning if *directory* does not match the **s_fname** field in the superblock of the filesystem to be mounted. The first six characters in the last component of *directory* are compared with the name in **s_fname** (i.e., mounting a filesystem named **spool** on **/usr/spool** won't cause a warning message, but mounting the same filesystem on **/mnt** will.).

Busy file structures cannot be dismounted with *umount*. A file structure is busy if it contains an open file or some user's working directory.

Notes

Only the super-user can use the *mount* command.

Some degree of validation is done on the file structure, however it is generally unwise to mount corrupt file structures.

Be warned that when in single-user mode, the commands that look in **/etc/mnttab** for default arguments (for example *df*, *ncheck*, *quot*, *mount*, and *umount*) give either incorrect results (due to a corrupt **/etc/mnttab** from a non-shutdown stoppage) or no results (due to an empty mnttab from a *shutdown* stoppage).

When multi-user, this is not a problem; the **/etc/rc2** scripts initialize **/etc/mnttab** to contain only **/dev/root** and subsequent mounts update it appropriately.

The *mount*(ADM) and *umount*(ADM) commands use a lock file to guarantee exclusive access to **/etc/mnttab**. The commands which just read it (those mentioned above) do not, so it is possible that they may hit a window, which is corrupt. This is not a problem in practice since *mount* and *umount* are not frequent operations.

When mounting a file system on a floppy disk you need not use the same *directory* each time. However, if you do, the full pathnames for the files are consistent with each use.

Always **unmount** filesystems on floppy disks before removing them from the floppy drive. Failure to do so requires running **fsck** the next time the disk is **mounted** .

The directory **/etc/fscmd.d/***TYPE* contains programs for each file system type; *mount*/**umount** invokes the appropriate binary.

Standards Conformance

mount is conformant with:

AT&T SVID Issue 2, Select Code 307-127;
and The X/Open Portability Guide II of January 1987.

mountall, umountall

mount, unmount multiple file systems

Syntax

/etc/mountall [**-**] [filesystem-table] . . .
/etc/mountall [**-a**]
/etc/umountall [**-k**]

Description

These commands can be executed only by the super-user.

The *mountall* command is used to mount filesystems according to a *filesystem-table*. (**/etc/default/filesys** is the default filesystem table.) The special file name "-" reads from the standard input.

Before each file system is mounted, it is checked using *fsstat*(ADM) to see if it appears mountable. If the file system does not appear mountable, it is checked, using *fsck*(ADM), before the mount is attempted.

mountall is called with the **-a** when the system autoboots. The **-a** flag causes output messages to be written to the file **/etc/bootlog**, and later mailed to the system administrator (see *boot*(HW)).

The *umountall* command causes all mounted file systems except **root** to be unmounted.

Files

Filesystem-table format:

column 1	block special file name of filesystem
column 2	mount-point directory
column 3	"-r" if to be mounted read-only; "-d" if remote
column 4	(optional) filesystem type string
column 5+	ignored

White space separates columns. Lines beginning with "#" are comments. Empty lines are ignored.

A typical filesystem-table might read:

/dev/dsk/0s1 /usr -r S51K

See Also

boot(HW), fsck(ADM), fsstat(ADM), mount(ADM), signal(S), filesys(F)

Diagnostics

No messages are printed if the filesystems are mountable and clean.

Error and warning messages come from *fsck*(ADM), *fsstat*(ADM), and *mount*(ADM).

Notes

The information displayed in Column 3 will only appear if the file system was mounted as a read-only or remote resource.

mvdir

moves a directory

Syntax

/etc/mvdir dirname newdirname

Description

mvdir moves directories within a file system. The directory (*dirname*) must be a directory. If there is already a directory or file with the same name as *name*, *mvdir* fails.

Neither name may be a sub-set of the other. For example, you cannot move a directory named **/x/y** to **/x/y/z**, and vice versa.

Notes

You must be *root* to use *mvdir*.

See Also

mkdir(C)

Standards Conformance

mvdir is conformant with:

AT&T SVID Issue 2, Select Code 307-127.

ncheck

generates names from inode numbers

Syntax

ncheck [**-i** numbers] [**-a**] [**-s**] [filesystem]

Description

ncheck with no argument generates a pathname and inode number list of all files on the set of file systems specified in **/etc/mnttab**. The two characters "/." are appended to the names of directory files.

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.

A single *filesystem* may be specified rather than the default list of mounted file systems.

Files

/etc/mnttab

See Also

fsck(ADM), sort(C)

Diagnostics

When the file system structure is improper, ?? denotes the "parent" of a parentless file and a pathname beginning with ... denotes a loop.

Notes

See *Notes* under *mount*(ADM).

The directory **/etc/fscmd.d/***TYPE* contains programs for each file system type; *ncheck* invokes the appropriate binary.

netutil

administers the micnet network

Syntax

netutil [option] [**-x**] [**-e**]

Description

The *netutil* command allows the user to create and maintain a network of UNIX machines. A Micnet network is a link through serial lines of two or more systems. It is used to send mail between systems with the *mail*(C) command, transfer files between systems with the *rcp*(C) command, and execute commands from a remote system with the *remote*(C) command.

The *netutil* command is used to create and distribute the data files needed to implement the network. It is also used to start and stop the network. The *option* argument may be any one of **install, save, restore, start, stop**, or the numbers 1 through 5 respectively. The **-x** option logs transmissions and the **-e** options logs errors. The **-x** and **-e** options work only when they are used in conjunction with **start, stop** or their decimal equivalents (4 and 5).

The **install** option interactively creates the data files needed to run the network. The **save** option saves these files on floppy or hard disks, allowing them to be distributed to the other systems in the network. If you save the micnet files to the hard disk, you can then use *uucp*(C) to transfer the files to the other machines. This option specifies the name of the backup device and prompts for whether this is the desired device to use. The user can specify an alternate device, including a file on the hard disk. The name of the default backup device is located in the file **/etc/default/micnet**. This can be changed depending on system configuration. The **restore** option copies the data files from floppy disk back to a system. The **start** option starts the network. The **stop** option stops the network. An *option* may also be any decimal digit in the range 1 to 5. If invoked without an *option*, the command displays a menu from which to choose one. Once an option is selected, it prompts for additional information if needed.

A network must be installed before it can be started. Installation consists of creating appropriate configuration files with the **install** option. This option requires the name of each machine in the network, the serial lines to be used to connect the machines, the speed of transmission for each line, and the names of the users on each machine. Once created, the files must be distributed to each computer in the network with the **save** and **restore** options. The network is started by using the

start option on each machine in the network. Once started, mail and remote commands can be passed along the network. A record of the transmissions between computers in a network can be kept in the network log files. Installation of the network is described in the *System Administrator's Guide*.

Files

/bin/netutil
/etc/default/micnet

See Also

mail(C), micnet(F), remote(C), rcp(C), systemid(F), top(F)

Value Added

netutil is an extension of AT&T System V provided in Altos UNIX System V.

nictable

process NIC database into channel/domain tables

Syntax

nictable -[**CDT**] [**-d** domain] [**-s** service] [**-t** transport]

Description

nictable is the tool responsible for taking the hosts.txt table supplied by the SRI Network Information Center and creating domain and channel tables.

The **-C** option causes the program to generate a channel table on the standard output. The **-D** option creates a domain table. This option should be combined with the **-d** option explained below to state which domain table you are building. The **-T** option creates a "top" or "rootdomain" table. No trailing domain spec is removed from the LHS entry.

There are several options for further restricting the number of hosts chosen. The **-d** domain option states that only hosts in the domain specified should be output. An exception to this is when **-d** is combined with **-T**. In this case, all entries will be output EXCEPT for those in the domain specified. The intention is that you grap all of one domain with **-D**, and then grab everybody else with **-T**. The **-s** service option states that only hosts that are listed as supporting the service specified should be output. The **-t** transport option is like **-s** except it states that only hosts supporting the transport protocol specified should be considered.

Typical usage involves two or three invocations:

> nictable -C < /etc/hosts.txt > smtpchannel
>
> nictable -D -d ARPA < /etc/hosts.txt > arpadomain

(and optionally)

> nictable -T -d ARPA < /etc/hosts.txt > rootdomain

Value Added

nictable is an extension of AT&T System V provided in Altos UNIX System V.

nlsadmin

network listener service administration

Syntax

nlsadmin -x
nlsadmin [options] net_spec

Description

nlsadmin administers the network listener process(es) on a machine. Each network has a separate instance of the network listener process associated with it; each instance (and thus, each network) is configured separately. The listener process "listens" to the network for service requests, accepts requests when they arrive, and spawns servers in response to those service requests. The network listener process will work with any network (more precisely, with any transport provider) that conforms to the transport provider specification.

The listener supports two classes of service: a general listener service, serving processes on remote machines, and a terminal login service, for terminals connected directly to a network. The terminal login service provides networked access to this machine in a form suitable for terminals connected directly to the network. However, this direct terminal service requires special associated software, and is only available with some networks (for example, the AT&T STARLAN network).

nlsadmin can establish a listener process for a given network, configure the specific attributes of that listener, and start and kill the listener process for that network. *nlsadmin* can also report on the listener processes on a machine, either individually (per network) or collectively.

The following list shows how to use *nlsadmin.* In this list, *net_spec* represents a particular listener process. Specifically, *net_spec* is the relative path name of the entry under **/dev** for a given network (that is, a transport provider). Changing the list of services provided by the listener produces immediate changes, while changing an address on which the listener listens has no effect until the listener is restarted. The following combination of *options* can be used.

no options — will give a brief usage message.

-x — will report the status of all of the listener processes installed on this machine.

net_spec will print the status of the listener process for *net_spec*.

-q *net_spec* will query the status of the listener process for the specified network, and will reflect the result of that query in its exit code. If a listener process is active, *nlsadmin* will exit with a status of 0; if no process is active, the exit code will be 1; the exit code will be greater than 1 in case of error.

-v *net_spec* will print a verbose report on the servers associated with *net_spec*, giving the service code, status, command, and comment for each. It also specifies the **uid** the server will run as, and the list of modules to be pushed, if any, before the server is started.

-z *service_code net_spec*
will print a report on the server associated with *net_spec* that has service code *service_code*, giving the same information as in the **-v** option.

-q -z *service_code net_spec*
will query the status of the service with service code *service_code* on network *net_spec*, and will exit with a status of 0 if that service is enabled, 1 if that service is disabled, and greater than 1 in case of error.

-l *addr net_spec* will change or set the address on which the listener listens (the general listener service). This is the address generally used by remote processes to access the servers available through this listener (see the **-a** option, below). *addr* is the transport address on which to listen and is interpreted using a syntax that allows for a variety of address formats. By default *addr* is interpreted as the symbolic ASCII representation of the transport address. An *addr* preceded by a **\x** will let you enter an address in hexadecimal notation. Note that *addr* must appear as a single word to the shell and must be quoted if it contains any blanks.

If *addr* is just a dash (-), *nlsadmin* will report the address currently configured, instead of changing it.

A change of address will not take effect until the next time the listener for that network is started.

-t *addr net_spec* will change or set the address on which the listener listens for requests for terminal service, but is otherwise similar to the **-l** option above. A terminal service address should not be defined unless the appropriate remote login software is available; if such software is available, it must be configured as service code 1 (see the **-a** option, below).

-i *net_spec* will initialize or change a listener process for the network specified by *net_spec*; that is, it will create and initialize the files required by the listener. Note that the listener should only be initialized once for a given network, and that doing so does not actually invoke the listener for that network. The listener must be initialized before assigning addressing or services.

[-m] -a *service_code* **[-p** *modules***] [-w** *id***] -c** *cmd* **-y** *comment net_spec*
will add a new service to the list of services available through the indicated listener. *service_code* is the code for the service, *cmd* is the command to be invoked in response to that service code, comprised of the full path name of the server and its arguments, and *comment* is a brief (free-form) description of the service for use in various reports. Note that *cmd* must appear as a single word to the shell, so if arguments are required, the *cmd* and its arguments must be surrounded by quotes. Similarly, the *comment* must also appear as a single word to the shell. When a service is added, it is initially enabled (see the **-e** and **-d** options below).

If the **-m** option is specified, the entry will be marked as an administrative entry. Service codes 1 through 100 are reserved for administrative entries, which are those that require special handling internally. In particular, code 1 is assigned to the remote login service, which is the service automatically invoked for connections to the terminal login address.

The **-m** option used with the **-a** option indicates that special handling internally is required for those servers added with the **-m** set. This internal handling is in the form of code embedded on the listener process.

If the **-p** option is specified, then *modules* will be interpreted as a list of STREAMS modules for the listener to push before starting the service being added. The modules are pushed in the order they are specified. *modules* should be a comma-separated list of modules, with no white space included.

If the **-w** option is specified, then *id* is interpreted as the user name from **/etc/passwd** that the listener should look up. From the user name, the listener should obtain the user ID, the group ID, and the home directory for use by the server. If **-w** is not specified, the default is to use the user ID **listen.**

A service must explicitly be added to the listener for each network on which that service is to be available. This operation will normally be performed only when the service is installed on a machine, or when populating the list of services for a new network.

-r *service_code net_spec*
will remove the entry for the *service_code* from that listener's list of services. This will normally be performed only in conjunction with the de-installation of a service from a machine.

-e *service_code net_spec*
-d *service_code net_spec*
will enable or disable (respectively) the service indicated by *service_code* for the specified network. The service must have previously been added to the listener for that network (see the **-a** option above). Disabling a service will cause subsequent service requests for that service to be denied, but the processes from any prior service requests that are still running will continue unaffected.

-s *net_spec*
-k *net_spec*
will start and kill (respectively) the listener process for the indicated network. These operations will normally be performed as part of the system startup and shutdown procedures. Before a listener can be started for a particular network, it must first have been initialized, and an address must be defined for the general listener service (see the **-i** and **-l** options, above). When a listener is killed, processes that are still running as a result of prior service requests will continue unaffected.

The listener runs as user ID **root**, with group ID sys. A special ID, user ID **listen** and group ID **adm**, should be entered in the **/etc/passwd** file as a default ID for servers. The listener always uses as its home directory **/usr/net/nls**, which is concatenated with *net_spec* to determine the location of the listener configuration information for each network. The home directory specified in the **/etc/passwd** entry for **listener** will used by servers that run as ID **listen.**

nlsadmin may be invoked by any user to generate reports, but all operations that affect a listener's status or configuration are restricted to the super-user.

Diagnostics

If the command is not run under the proper ID, an error message will be sent to standard error and the command will terminate.

Files

/usr/net/nls/*net_spec*

See Also

Network Programmer's Guide

pcu

port configuration utility

Syntax

pcu
pcu -c
pcu [**-q**] **-d** port ...
pcu [**-q**] **-a** port ...
pcu [**-q**] [**-t** term] [**-x**] [**-b** baud] port ...
pcu [**-q**] [**-m** modem] [**-x**] [**-b** baud] [**-s** sec] port ...
pcu [**-q**] [**-p** printer] [**-b** baud] [**-n** name] [**-i** model] port

Description

The *pcu* command lets the system administrator configure the tty ports on a system.

When adding a new tty controller board to a system, the system administrator should perform the following:

- Install the board.
- On EISA systems, run the EISA configuration utility (supplied on the ECU diskette that accompanies your system).
- Run the *uconfig*(ADM) command, and reboot your system.
- Create an entry for the controller board in the file **/etc/card_info**, as described in *card_info*(F).
- Run *pcu*(ADM).

Information Mode

If used with the **-c** option, *pcu* will display the current port assignments with respect to the EISA boards in the system.

Visual Mode

If the user supplies no arguments, *pcu* will enter visual mode, allowing the user to interactively configure ports to contain a variety of devices. A window displays the current state of each board, or an eight port section of a board (if it contains more than eight ports). Each line in this window gives information about one port. Various fields on this line describe the type of device connected to a port; its baud rate, model, and other information as applicable. As the user interacts with *pcu*, the data displayed in this window will be updated in an immediate, "real-time" manner. The changes you propose are displayed in the *pcu* window; however, the actual changes are not implemented until you press the Save screen labeled key (F5).

Batch Mode

If the user supplies arguments, *pcu* will execute the requested function on one or more specified ports in batch mode. This mode is less flexible than interactive visual mode in that only one type of command may be given at a time. Changes take effect immediately, and the user does not receive any display of the current state of the system's configuration. Batch mode exists so that ports may be added from shell-scripts or by users who know exactly what they want to change and prefer to do so from the command line.

Options

-c The current port assignments for the EISA boards in the system are displayed.

The following options specify the action and the device type. Other options exist to specify the baud rate, timeout, transparent printer attachment, printer name, or printer model. If these are missing, they are assigned default values appropriate for the type of each device. These default values are set up by the system administrator when *pcu* is installed.

-d The inittab entries for the specified ports will be disabled. With the exception of **-q**, no other options may be specified with the **-d** option.

-a The inittab entries for the specified ports will be enabled. With the exception of **-q**, no other options may be specified with the **-a** option.

-t *term* The specified ports will be changed to support a terminal of type *term* and their **/etc/inittab** entries will be set to **respawn**.

-m *modem* The specified ports will be changed to support a modem of type *modem* and their **/etc/inittab** entries will be set to **respawn**. Modem may be one of **in**, **out**, and **bi** for **dial-in, dial-out**, and **bidirectional**, respectively.

-p *printer* The specified port will be changed to support a printer of type *printer* and its inittab entry will be set to **respawn**. *printer* may be either **Parallel** or **Serial**. Note that only one port may be specified when this option is given.

The following options may be specified in conjunction with **-t,-m**, or **-p**, or they may appear alone (or in combination with each other) to set the appropriate fields without changing the device type:

-b *baud* Sets the baud rate of the specified ports to *baud*, where *baud* is a valid gettydefs label. For a printer port, *baud* should be the numeric baud rate value (such as 9600).

-s *sec* Sets the number of seconds for the timeout option to getty. The **-s** option is not applicable to printer ports.

-n *name* Sets the printer name to *name*. The **-n** option is only applicable to printer ports.

-i *model* Sets the printer model to *model*. The **-i** option is only applicable to printer ports.

-x Attaches a transparent printer to the specified terminal or modem port. The **-x** option is not applicable to printer ports. For terminal ports, the options **-n** and **-i** may also be supplied, provided that only one port is specified on the command line. A printer is not created for modem ports. The user will invoke *xprcat*(C) to use the transparent printer on modem ports. For terminal ports, the default for the **-i** option is the standard *lp* interface script. The default printer's name is obtained by replacing ''tty'' with ''xpr'' in the port's ttyname (e.g., port tty128 will have a transparent printer name of xpr128).

-q The quiet option is used when *pcu* is to be run from a shell script. It suppresses any interaction with the user by assuming answers to questions that the user would ordinarily be asked. When the user is changing or deleting a printer that has requests pending on it, he would normally be asked if those requests should be saved or moved to another printer. With the **-q** option the answer is assumed to be **no** for both of these situations.

Visual Mode

At any time in visual mode, brief context-sensitive help information is available by pressing the question mark key (?) or HELP key (if your terminal is so equipped).

In visual mode, certain letter-keys may be substituted for function-keys and arrow-keys. For function-key substitution, use the letter-key corresponding to the capital letter in the function-key label displayed at the bottom of the *pcu* screen. For example, use the **e** letter-key as a substitute for the Enable function-key (F1), or use the **f** letter-key as a substitute for the deFault function-key (F2). For arrow-key substitution, the letter-keys **h**, **j**, **k**, and **l** may be used as substitutes for the left, down, up, and right arrow-keys, respectively.

In visual mode the screen is divided into three areas, which display different types of information. The first area is the backplane which is a pictorial representation of the back of the machine. Each slot is designated by a horizontal line. The currently displayed slot is highlighted in reverse video bold. The currently displayed slot may be changed with the Prevslot and Nextslot screen labeled keys (F7 and F8, respectively).

The second area is a window which displays the contents of the currently displayed slot. The title bar of this window identifies the slot number and the name of the EISA board in that slot. If the name of the board is not known, its uncompressed EISA identifier will be displayed instead. If the slot is empty, the word NO PORTS will be displayed. The interior of this window contains a line for each port on that board up to a maximum of eight ports. If the board contains more than eight ports, they may be paged through eight ports at a time by using the previous page and next page keys. Ports that are turned off in the inittab file will have their lines dimmed. Ports may be turned on or off via the Enable/Disable screen labeled key (F1).

The definition of F1 changes line by line. For lines whose inittab entries are already off, F1 invokes the Enable function to turn the entry on. If the port is already on, pressing F1 will invoke the Disable function to turn it off. Since the currently selected line displays in reverse video bold, the user would not be able to tell whether that port is on or off without moving off that line, were it not for the fact that the screen label of the F1 key contains this information. If the label for F1 is Disable, then the port is now on. If the label is Enable, then the port must now be off. This window will be empty for boards that are unknown (that is, for boards that have no corresponding entry in the **/etc/card_info** file) and for empty slots. If the window contains any lines, there will always be a currently selected line which is displayed in reverse video bold.

The currently selected line may be changed by using the up and down arrow keys. If the bottom or top of the window is reached and a down or up arrow key is pressed, the previous page or next page, if any, will be displayed. The currently selected line will be the first or last line of the new page.

Lines may also be marked for use with commands which can affect multiple lines. Lines are marked by pressing the Return key when the entire currently selected line is displayed in reverse video bold (that is, no fields are selected—see below for a discussion of fields). Marked lines are displayed in reverse dim.

After marking a line the currently selected line is advanced to the next line, unless the marked line happens to represent the last port on the current board. Lines may be unmarked by pressing return again while the marked line is the currently selected line and no fields are selected.

The Enable and Disable commands (F1 key) may be used to enable or disable all marked lines. If no lines are marked, the command only applies to the currently selected line.

Note that the command shown on the screen label of the F1 key is the command applied to all marked lines without regard to whether the ports associated with those marked lines are currently on or off. This command does not simply toggle the state of the port for each marked line. Sometimes two or more **/dev** entries are aliases which control the same hardware port. An example of this is **/dev/tty1A** and **/dev/tty1a**, which both control the COM1 port: the first with modem control, the second without. No more than one inittab entry for aliased ports may be active at once. *pcu* ensures this by disallowing the user from activating more than one aliased entry at a time.

The deFault command (F2 key) is used to select or deselect the system default printer. The screen label for the F2 key indicates for each printer device whether it is currently the system default printer. It does this by prefixing the command name, deFault, with an asterisk. Pressing F2 when the currently selected line is displaying the system default printer deselects it. The printer is unchanged; it is simply no longer the system default printer. If a Save command is done at this point, then a system default printer is not established.

Pressing F2 when the currently selected line is displaying a printer that is not currently the system default printer selects that printer as the new system default printer. Only one system default printer may exist on any system, so selecting a system default printer deselects any other printer which was previously the system default printer. When the currently selected line does not contain a printer, the screen label for the F2 key will be blank and pressing F2 will have no effect.

The Copy command (F3 key) copies the entry on the currently selected line to all marked lines.

The Undo command (F4 key) will undo the proposed changes to any marked lines by setting them to the state they were in at the time of the last Save command. If no Save command has been given during the current invocation of *pcu*, these lines are set to the values they had when *pcu* was invoked. If the Undo command is given and no lines are marked, then any proposed changes to the currently selected line are undone. The shifted Undo command (shifted F4 key) will undo all changes made to all ports on all board since the last Save or the beginning of this invocation of *pcu*.

Any changes that the user makes in *pcu* are proposed changes until the user issues a Save command by pressing the Save screen labeled key (F5) or shift F5, which means save and quit. When *pcu* saves changes, it will update **/etc/inittab** and execute an 'init q' command. In addition, it will kill any gettys or uugettys that are running on changed ports. This causes these ports to respawn the getty or uugetty, thus effecting any changes to that port, provided that no user is currently logged in to that port. If a user is logged on to a port while *pcu* is changing it, the changes will take effect when the user logs off.

The user may quit without saving changes by pressing the Quit key (F6) or shift F6. Pressing F6 will cause *pcu* to first check to see if any changes have been made that have not yet been saved. If *pcu* finds that changes are pending, it will ask for confirmation. Answering with a 'y' or 'Y' (or any response beginning with these letters) followed by a carriage return will cause *pcu* to exit. Any other response will be taken to mean 'no' and *pcu* will continue. To exit immediately without saving changes or incurring the confirmation message, press shift F6.

A screen redraw key (ctrl-L) is provided in case the display is overwritten by other users writing to your terminal or messages sent to the console.

Each line is divided into several fields which display information about the corresponding port. The user proposes changes by altering the values of these fields; however, no changes take effect until the user issues the Save, or Save and Quit commands described above. Fields are selected by using the left and right arrow keys. The currently selected field will be displayed in reverse dim. Pressing return while a field is selected will bring up the pop-up menu for that field. Menu selections may be made by using the up arrow and down arrow keys to highlight the desired menu item, then pressing return to select that menu item and close the menu. To exit a menu without selecting a menu item, press the escape key. Menus may have more than one page. The previous page and next page keys may be used to navigate from one menu page to another. In addition, scrolling past the bottom or top of a menu page will display the next or previous page. Note that navigation in a multipage menu is the same as navigation in a multipage window.

In addition to normal menu selection, some menus allow the user to type in a value to be returned by the menu. Such menus have a blank line at the bottom of each menu page which cannot be highlighted. Any keys other than the arrow keys, escape key, or return key which are pressed by the user will be echoed in this blank line and be returned as the selected item when return is pressed. Any item so returned will be subject to validation by *pcu*. *pcu* may reject the choice or simply warn the user that some action outside of *pcu* must be taken in order for the choice to have the desired effect.

An a example of this is when the user selects a terminal type that does not appear in the menu. *pcu* will check this selection to ensure that an entry by that name has been compiled into the appropriate **/usr/lib/terminfo/***?* **/** directory. If no compiled entry is found, *pcu* will inform the user that an entry for that terminal must be compiled with *tic* after exiting *pcu*.

When typing in a new menu item, some basic line editing capabilities are provided. The backspace key will delete the last character and move the cursor back one space. The left arrow key will move the cursor back one space without erasing characters it passes over. The right arrow key will move the cursor one space to the right without affecting characters it passes over. Text typed at the cursor will be inserted at that point and characters to the right of the cursor will be pushed further to the right. The delete line key may be used to delete all characters on the line and place the cursor on the first character of the line. The up arrow and down arrow keys and the previous and next page keys retain their normal meanings. Selecting one of these keys while inputing a new menu item aborts the new item. As usual, the escape key can be used to cancel the menu request.

If the information in a field is too long to be displayed in that field, the shifted right and left arrow keys may be used to scroll horizontally by the width of the field to display the remaining text. When the user moves off of such a field the field will automatically redisplay the initial portion of its text. In *pcu*, only the baud rate field takes advantage of this capability. A baud rate is actually a sequence of baud rates to be tried one after the other by getty or uugetty when a user tries to log-in. These baud rate sequences are defined as chains of entries in the file **/etc/gettydefs**. The baud rate menu displays the baud rates available to a device as the gettydef label of the head of a baud rate sequence followed by a colon, followed by a list of baud rates, one for each gettydef entry in the chain. The text of a baud rate field is set up such that the shifted right or left arrow keys will display the next or previous baud rate in the sequence.

The first field of each line is an unselectable field which displays the name of the port it describes. The name of the port is preceded with a colon (:) if the port is currently disabled. The second field displays the type of the device. For terminals, this is the terminfo name for that type of terminal. For modems, the type field may be 'Dial-in', communication. Dial-in modems use getty to allow users to log in remotely, but the computer is not able to dial out through them. Dial-in/out modems use uugetty to allow for remote user log-in and to allow the computer to dial out to other computers. Dial-out modems have their ports disabled so that users cannot log in but they may be used to call out to other computers. For a printer, the type field may be 'Serial' or 'Parallel.'

The third field, the device field, determines the type of device connected to the port. It may be a terminal, a printer, a modem, or a terminal or model with a transparent printer attached. Whenever the device field is changed the remaining fields in the line will take on the default values for that type of device as set up by the system administrator (see ''Configuration'' below).

The fourth field is the baud rate field. The baud rate menus may contain different entries for each device type if the system administrator deems some baud rate chains in **/etc/gettydefs** to be more appropriate for modems than terminals, for example. This configurability can be used to reduce the number of menu items in a particular baud rate menu (see "Configuration" below).

The fifth field contains flags to pass to the getty or uugetty commands or the name of the printer attached to the port, depending on the type of device as selected by field three. For printers, it contains the name of the printer. The sixth field is only used for printers. It contains the name of the model file that contains the lp interface script for that type of printer.

The third area of the screen is the Terminal Cluster Unit (TCU) area. The TCU area is located below the ports window and displays a pictorial representation of the TCUs that hang off of a multidrop card. This area only displays TCUs if the currently selected slot contains an MDC/2 (EISA Multidrop Card).

TCUs are chained together in increasing order of address. The currently selected TCU will be highlighted, with its dip switch settings, specifing the upper five bits of its address, displayed as a binary number. The collection of TCUs may be traversed, a TCU at a time, by using the previous page and next page keys. Also the tab key (read TCU Advance Block in *pcu*) may be used to jump over blocks of TCUs. The currently selected TCU and the corresponding page in the port window will always match. Changing a page in the port window will cause the next or previous TCU in the TCU area to be selected. Changing the currently selected TCU with the previous page, next page, or tab keys will cause the corresponding page in the port window to be displayed.

Security and Concurrency

pcu may be run by any user; however, only root and the system administrator are allowed to make changes. *pcu* may be run from either single-user or multiuser mode. Multiple copies may be run concurrently, but only one of these copies will be permitted to make changes at any given time. When multiple copies of *pcu* are running, the first user to make any proposed changes will obtain a lock. All other copies of *pcu* will be prevented from making changes until the copy that owns the lock has exited. All other copies will function normally except that any attempt to make changes will result in the following error message:

```
Others are changing ports -- your pcu is currently read-only
```

When the other *pcu* user exits your *pcu* will again be able to make changes, but a noticeable delay will occur while *pcu* re-initializes itself. Also, if a *pcu* is started while another is in the process of writing a needed file there may be a delay in initialization as the first user will have obtained a lock on that file.

Users who have write permission for **/etc/inittab** may make changes to this file, and *pcu* will recognize these changes if a few simple conventions are followed. First, *pcu* uses the comment, #MODEM, at the end of the entry for a modem in order to avoid imposing an arbitrary naming convention on the baud rate labels in **/etc/gettydefs**. A user editing the **/etc/inittab** file to add a modem should include the comment, **#MODEM**, at the end of the entry to inform *pcu* that the new entry describes a modem device. Second, *pcu* assumes that all non-transparent printers will run the command 'lpset (Parallel | Serial) tty baud'. Third, any port names added by editing **/etc/inittab** must match the *regex*(S) pattern in the **/etc/card_info** entry (see the description of **/etc/card_info** below) for the board on which they occur, assuming that such an entry exists in **/etc/card_info** for this board or subfunction. Note that since **/etc/inittab** is built from files in **/etc/conf/init.d** concatenated to **/etc/conf/cf.d/init.base**, any changes made to **/etc/inittab** will disappear when an new driver is installed or the next time any user of *pcu* saves changes.

Configuration

The system administrator customizes the behavior of *pcu* by editing the files in the **/etc/PCU/defaults** directory. The defaults for each device type are given in **/etc/PCU/defaults/defaults**. The **/etc/gettydefs** labels that are displayed in the baud rate menu for each device type are given in the files **/etc/PCU/defaults/*_labs**. The printer models to be displayed in the models menu are given in **/etc/PCU/defaults/models**. The names of terminals that are to be displayed in the type menu for terminals are given in **/etc/PCU/defaults/term_type**.

The **/etc/card_info** file contains information on each board that *pcu* will use to determine the name of the board, the types of the ports on the board, the number of ports on the board, the driver for that board, the assignment of minor numbers, the name pattern for the **/dev** entries. Each entry contains up to seven fields which are separated by colons.

The first field of a **card_info** entry contains the uncompressed EISA identifier for the board (the minor revision level may be omitted). It is used to look up the entry for a specific board. For ISA tty controller boards, field one contains the word **ISA**. The second field contains the type of the ports on the board or EISA subfunction described by this entry. The third entry contains the full name of the board which is displayed for the user in the title bar of the ports window in *pcu*.

The fourth field contains a *regex*(S) pattern to identify the ports that belong to this board or subfunction. This regex pattern returns a value (usually numeric) that distinguishes one port on the board from the next (for example, if the pattern tty([0-9]{2,4})$0 was given the port name tty03, the 03 would be returned and regex would succeed).

The fifth field contains a *printf*(S) format string for printing a port's ttyname, and takes as it's single argument the value returned by the regex pattern. For example, the format string **tty%02d** would be used with the regex pattern example given above.

The sixth and seventh fields are only used by ISA board entries. The sixth field contains a range of numbers of the form *n1-n2* used as arguments to the format string in field five to generate ttynames for the ports attached to this board. For EISA entries, *pcu* gets the port range for **/etc/slot_info**, which is created by *uconfig*(ADM) and its execution of **pack.d/*/*node** scripts. Field seven contains the device driver device name for this board as found in the first field of the device driver's *mdevice*(F) file entry.

Files

/etc/inittab
/etc/xprtab
/etc/gettydefs
/etc/card_info
/etc/slot_info
/etc/PCU/pcu.lockfile
/etc/PCU/init.d/*
/etc/PCU/defaults/*

See Also

uconfig(ADM), idmkinit(ADM), lpadmin(ADM), uugetty(ADM), tic(C), regcmp(S), regex(S), inittab(F), gettydefs(F), getty(M), terminfo(M), eisacfg(HW)

Owner's Guide (for description of EISA Configuration Utility)

Diagnostics

pcu will halt with an exit status of 1 if an error is encountered. An exit status of 0 implies success.

Notes

pcu requires that the TERM environment variable be set up correctly. If this is not the case, *pcu* may garble the display and render the function keys unusable. Since the Quit key could be affected, *pcu* allows a second way to quit without saving changes. If the BREAK key is pressed, an interrupt will be sent to *pcu* which causes it to quit immediately.

Different terminals support different video attributes. *pcu* assumes that the terminal on which it is run will have at least one of bold or dim, and at least one of reverse dim or reverse bold. A selected field cannot be highlighted if the second condition is not met, forcing the user to count the number of right arrow keystrokes given (or activate and cancel a pop-up menu) to see which field the cursor is on. The violation of the second condition is a more serious impairment to the functionality of *pcu*.

pcu is only supported on the following Altos terminals: Altos V, Altos VII, Altos 6010, Altos 6160, and the system console (uses ansi terminfo entry). The IBM-style keyboard used by the system console cannot distinguish between shifted and unshifted left and right arrow keys. Users of *pcu* should use the angle bracket keys ('<' and '>') to represent the shifted left and right arrow keys. If a monochrome graphics card is used for the system console, *pcu* users will not be able to distinguish reverse dim from reverse bold (see above), since these cards support only straight reverse video.

Value Added

pcu is an extension of AT&T System V provided in Altos UNIX System V.

profiler: prfld, prfstat, prfdc, prfsnap, prfpr

UNIX system profiler

Syntax

/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

The *prfld*, *prfstat*, *prfdc*, *prfsnap*, and *prfpr* routines form a system of programs to facilitate an activity study of the operating system.

The *prfld* program 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*.

The *prfstat* program 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.

The *prfdc* and *prfsnap* programs 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*.

The *prfpr* program 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*.

Files

/dev/prf	interface to profile data and text addresses
/unix	default for system namelist file

proto

prototype job file for at, cron and batch

Syntax

/usr/lib/cron/.proto

/usr/lib/cron/.proto.*queue*

Description

When a job is submitted to *at*(C) or *batch*(C), the job is constructed as a shell script. First, a prologue is constructed, consisting of:

- A header whether the job is an *at* job or a *batch* job (actually, *at* jobs submitted to all queues other than queue **a**, not just to the batch queue **b**, are listed as *batch* jobs); the header will be

  : at job

  for an *at* job, and

  : batch job

  for a *batch* job.

- A set of Bourne shell commands to make the environment (see *environ*(5)) for the *at* job the same as the current environment;

- A command to run the user's shell (as specified by the SHELL environment variable) with the rest of the job file as input.

At then reads a prototype file, and constructs the rest of the job file from it.

Text from the prototype file is copied to the job file, except for special variables that are replaced by other text:

$d is replaced by the current working directory
$l is replaced by the current file size limit (see *ulimit*(2))
$m
is replaced by the current umask (see *umask*(2))
$t is replaced by the time at which the job should be run, expressed as seconds since January 1, 1970, 00:00 Greenwich Mean Time, preceded by a colon

$< is replaced by text read by *at* from the standard input (that is, the commands provided to *at* to be run in the job)

If the job is submitted in queue *queue*, *at* uses the file **/usr/lib/cron/.proto.***queue* as the prototype file if it exists, otherwise it will use the file **/usr/lib/cron/.proto.**

Examples

The standard **.proto** file supplied is:

```
#ident "@(#)adm:.proto        1.2"
cd $d
ulimit $l
umask $m
$<
```

which causes commands to change the current directory in the job to the current directory at the time *at* was run, to change the file size limit in the job to the file size limit at the time *at* was run, and to change the umask in the job to the umask at the time *at* was run, to be inserted before the commands in the job.

Files

/usr/lib/cron/.proto
/usr/lib/cron/.proto.*queue*

See Also

at(C), sysadmsh(ADM), atcronsh(ADM)

rc0

run commands performed to stop the operating system

Syntax

/etc/rc0

Description

This file is executed at each system state change that needs to have the system in an inactive state. It is responsible for those actions that bring the system to a quiescent state, traditionally called "shutdown."

One system state requires this procedure: state 0 (the system halt state). Whenever a change to this state occurs, the */etc/rc0* procedure is run. The entry in **/etc/inittab** might read:

```
s0:0: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 system is shut down.

The recommended sequence for */etc/rc0* is:

Stop System Services and Daemons.

Various system services (such as Remote File Sharing 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* (ADM). 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.

Files

The execution by **/bin/sh** of any files in **/etc/shutdown.d** occurs in ASCII sort-sequence order. See *rc2* (ADM) for more information.

See Also

killall(ADM), rc2(ADM), shutdown(ADM)

rc2

run commands performed for multiuser environment

Syntax

/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 "multiuser" state.

The actions performed by */etc/rc2* are found in files in several directories and are executed in a prescribed order to ensure proper initialization. */etc/rc2* performs the following functions in the order in which they appear:

1. Runs the script */etc/conf/bin/idmkenv*. This script sets up the new kernel environment if a new kernel has been configured, calls *idmkinit* to rebuild the **/etc/inittab** file, and links files to the **/etc/idrc.d** and **/etc/idsd.d** directories to be run by */etc/rc2*.

2. Runs the system setup scripts in the directory **/etc/rc2.d**. Some of the scripts in this directory are front-end scripts to run other scripts in the subdirectories of **/etc/rc.d**.

3. Runs system setup scripts in the directory **/etc/rc.d**. This directory exists for XENIX compatibility. It contains subdirectories named with the numerals 0 to 9. Each subdirectory contains scripts that perform certain system startup functions (for example, the directory **/etc/rc.d/3** contains scripts that handle crash recovery). All of these scripts are run by the front-end scripts in **/etc/rc2.d**. Any other individual scripts in the directory are run.

4. Runs the system setup scripts in the directory **/etc/idrc.d**, which contains scripts from the driver packages linked from **/etc/conf/rc.d**.

5. Runs the scripts in **/etc/idsd.d**, which contains shutdown scripts linked from **/etc/conf/sd.d**.

6. Runs the script **/etc/rc**. This script exists for XENIX compatibility. It is an empty file, but you can add initialization commands to the file. These commands are run last during the initialization.

The setup scripts are executed by **/bin/sh** in ASCII sort-sequence order (see Files for more information). When functions are added that need to be initialized when the system goes multiuser, an appropriate file should be added in **/etc/rc2.d**.

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.

MOUNTFSYS

```
#    Set up and mount file systems
cd /
/etc/mountall
```

uucp

```
#    clean-up uucp locks, status, and temporary files
rm -rf /usr/spool/locks/*
```

/etc/rc2 also sets certain environment variables, including the **TZ** variable by reading */etc/TIMEZONE*, thus establishing the default environment for all commands that follow.

Files

Here are some hints about files in **/etc/rc.d**:

The order in which files are executed is important. Since they are executed in ASCII sort-sequence order, the first character of the file name is a sequence indicator that helps keep the proper order. Thus, files starting with the following characters would run accordingly:

[0-9]	very early
[A-Z]	early
[a-n]	later
[o-z]	last

Files in **/etc/rc.d** that begin with a dot (.) will not be executed. This feature can be used to hide files that are not to be executed for the time being without removing them. The command can be used only by the super-user.

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 beginning with other characters are ignored.

See Also

shutdown(ADM), init(M), "Starting and Stopping the System" chapter of the *System Administrator's Guide*

reduce

perform audit data analysis and reduction

Syntax

reduce [-s session] [**-p** selection file]

Description

reduce performs selective audit data reduction on compacted audit output files which were written by the audit daemon. Each audit record from the compaction files is examined during reduction to see if it meets the selectivity criteria established by the Audit Administrator. If so, the record is formatted and output to standard output.

Reduction is performed on all files written by the audit daemon during a specified boot *session.* Each time the Audit subsystem is enabled and disabled, a new session number is generated and this is used to stamp the filenames generated during that session so that they are easily recognizable. The audit daemon records each filename that it writes compacted data to in a log file. The log file is always written to the secure directory, **/tcb/files/audit.** Each session log file is uniquely named with the prefix **CAFLOG.** followed by the session number. Thus by specifying a session number for reduction, *reduce* is able to locate the log file and read it to determine certain setup parameters and the list of input files to be reduced.

Data is reduced based on a set of input selection criteria that governs the selection of records for printing. Records may be selected based on event types, time of event occurrence, user ID of record, group ID of record, or by specific object type. To selectively reduce, *auditsh*(ADM) is used to set up the audit selection file. This file is then specified to *reduce* upon invocation. Time interval selection allows for records to be selected only if they occurred within a certain time period. Event type selection allows records to be selected only if the specified event type is desired. Both user ID and group ID selection allows records that were generated by certain users or groups to be selected. Lastly, object selection applies to those record types referring to a specific file. Some records refer to multiple files and a single match for those record types will result in the record being selected. Time and event type selection always takes precedence over user/group ID and object selection (e.g. if a record has an event type that is not selected but the user ID is, the record will be discarded). If a record is selected based on time and event type, if any of user ID, group ID, or object matches a field in the record, the record is selected. If only time and event types are specified, all records of matching event types in the interval are selected. If only event type selection is requested, all matching events are selected from every record produced in that session. (e.g. If the event mask enables selection for all events and no time interval is specified, all records will be output)

The format of the reduced data varies on the type of event being processed. Each record will include the process ID of the process being audited, the date and time of the event, the type of audit event, an indication of success or failure for the event, and if applicable, object names that were accessed.

Items that are displayed for events include the following:

Process ID — The process ID of the process that generated the audit record.

User IDs — The login user ID, effective user ID, real user ID, effective group ID, and the real group ID are output for the process generating the audit record.

Date/Time — Each audit record is time stamped at generation time. The time value is formatted to produce a date/time string similar to that printed by *ctime*(S).

Event Type — Each audit record is classified into a certain event depending on what type of system call was performed or what type of action was taken by a trusted application.

Action — Many event types are broad categories into which certain actions are classified. The reduction program makes use of other data in the record to provide further discrimination between process actions that fall into the category. For system calls, the actual system call audited is output. For applications, a more specific action identifier is provided.

Object(s) — Many events involve files or special devices which are classified as objects. The name of the objects affected by process actions are recorded for data reduction. Depending on the event and action type, some output records may include one or more object names.

Modes — For certain event types, the modes of a file or IPC object may be modified. For these records, the old and new values of the owner, group, and the object mode are displayed.

Username — Some events are user account oriented such as login and logoff as well as certain administrative functions. These output records include the username of the account that was responsible for the audited action.

Result — Each output record carries an indicator of whether the action was successful or not. Unsuccessful actions are sometimes more important that successful ones since they may indicate attempts to penetrate the system. For system calls that fail, the specific error number and error message is output. For applications, an error message describing the failure is output.

See Also

auditsh(ADM), auditd(ADM), audit(HW), ''Maintaining System Security,'' chapter of the *System Administrator's Guide*

Diagnostics

Upon successful completion, the program exits with status 0.

Value Added

reduce is an extension of AT&T System V provided in Altos UNIX System V.

relogin

rename login entry to show current layer

Syntax

/usr/lib/layersys/relogin [-s] [line]

Description

The *relogin* command changes the terminal *line* field of a user's *utmp*(F) entry to the name of the windowing terminal layer attached to standard input. *write*(C) messages sent to this user are directed to this layer. In addition, the *who*(C) command will show the user associated with this layer. The *relogin* command may only be invoked under *layers*(C).

relogin is invoked automatically by *layers*(C) to set the *utmp*(F) entry to the terminal line of the first layer created upon startup and to reset the *utmp*(F) entry to the real line on termination. It may be invoked by a user to designate a different layer to receive *write*(C) messages.

-s Suppress error messages.

line Specifies which *utmp*(F) entry to change. The *utmp*(F) file is searched for an entry with the specified *line* field. That field is changed to the line associated with the standard input. To learn what lines are associated with a given user, say **jdoe**, enter:

```
ps -f -u jdoe
```

and note the values shown in the **TTY** field [see *ps*(C)].

Files

/etc/utmp data base of users versus terminals

Diagnostics

Returns **0** upon successful completion, **1** otherwise.

See Also

layers(C), mesg(C), ps(C), who(C), write(C), utmp(F)

Notes

If *line* does not belong to the user issuing the *relogin* command or standard input is not associated with a terminal, *relogin* will fail.

removepkg

remove installed package

Syntax

removepkg [software_package]

Description

The *removepkg* command will remove the AT&T-style software package specified as an argument to *removepkg* or will remove the software package the user selects if no argument is given to *removepkg*.

If an argument is specified, *removepkg* will search the list of previously installed packages and remove the first name it matchs. If no name is matched, the user is given an error message.

If no argument is specified, *removepkg* will query the user, via a menu, which package to remove.

Notes

You must invoke *removepkg* on the console.

This command does not work on packages installed with **custom**(ADM).

See Also

displaypkg(ADM), installpkg(ADM)

restore

UNIX incremental filesystem backup restore

Syntax

restore [**-c**] [**-i**] [**-o**] [**-t**] [**-d** device] [pattern [pattern] ...]

Description

This utility acts as a front end to *cpio*(C), and thus reads *cpio* format tapes or floppies. This utility should only be used to restore backups made with the AT&T *backup*(ADM) utility, not *xbackup*(ADM).

-c complete restore. All files on the tape are restored.

-i gets the index file off of the medium. This only works when the archive was created using *backup*. The output is a list of all the files on the medium. No files are actually restored.

-o overwrite existing files. If the file being restored already exists it will not be restored unless this option is specified.

-t indicates that the tape device is to be used. MUST be used with the **-d** option when restoring from tape.

-d <*device*> is the raw device to be used. It defaults to **/dev/rdsk/f0q15d** (the 1.2M floppy).

When doing a restore, one or more patterns can be specified. These patterns are matched against the files on the tape. When a match is found, the file is restored. Since backups are done using full pathnames, the file is restored to its original directory. Metacharacters can be used to match multiple files. The patterns should be in quotes to prevent the characters from being expanded before they are passed to the command. If no patterns are specified, it defaults to restoring all files. If a pattern does not match any file on the tape, a message is printed.

When end of medium is reached, the user is prompted for the next media. The user can exit at this point by entering ``q''. (This may cause files to be corrupted if a file happens to span a medium.) In general, quitting in the middle is not a good idea.

If the file already exists and an attempt is made to restore it without the **-o** option, the file name will be printed on the screen followed by a question mark. This file will not be restored.

In order for multi-volume restores to work correctly, the raw device MUST be used.

See Also

sh(C)

rmail

submit remote mail received via UUCP

Syntax

rmail user ...

Description

rmail interprets incoming mail received via *uucp*(C), passing the processed mail on to *submit*(ADM) for processing by the MMDF mail system. *rmail* is explicitly designed for use with UUCP and the MMDF *submit* program. It is not intended for use by regular users.

Rmail performs several conversions on the incoming mail before calling *submit*. The conversions change addresses from the UUCP routing style (lists of hosts separated by the character '!') to the domain style of address used within the MMDF mail system. The incoming message is dealt with in the following manner:

1) The initial "From" (or ">From") line is processed to discover the originating site and the sender of the message. Some UUCP mailers do not supply this information as part of the message body. If the originating site cannot be found from this information, the program environment is inspected for the variable "ACCTSYS"; this is set to the originating system by some implementations of UUCP. The originating system is used as a table lookup value into the mmdf table "rmail.chans," the file contains site/channel pairs. If a match is found the resulting channel is used for the submit phase. The default UUCP channel is used if no match is found. The default channel name is specified in conf.c source and can be runtime tailored. Typically it is "uucp". The existence of this channel is MANDATORY to prevent dropping mail from unknown hosts.

2) The body of the message is inspected looking for any header lines containing addresses; the lines are "From:", "To:", "Cc:", "Bcc:" and "Sender:". By scanning the address chains, the addresses in these lines are converted into "user@known-site.domain" form using the MMDF tables to evaluate whether the mailer knows the site. For this to work properly, the unqualified name of all sites should exist in the appropriate domain tables. The scanning stops when an unknown site is discovered and a composite address will be created. The "From:" line is treated specially to preserve any comment information which may have been inserted by the originating mailer.

3) The 'Date:' line is also re-written into ARPA standard form.

Before *submit* is called, the message is re-written into RFC822/733 form with all addresses obeying the appropriate convention. Any missing header lines are supplied. The destination address for the message is taken from the argument to *rmail* , and so the header re-writing which is done does not affect the routing of the message.

See Also

mail(C), uucp(C), submit(ADM)

routines

finds driver entry points in a driver object module

Syntax

/etc/conf/cf.d/routines file.o ...

Description

routines searches each of the specified object modules for the names of routines used in the device driver and displays them on the screen

This script is used when installing a device driver with the Link Kit. Write down the names produced by *routines*, then sort through the names to determine the interrupt priority level (the highest number following the *spl* prefix) and the relevant configurable driver routines (the collection of routines with a common prefix and the suffixes *open, close, read, write, ioctl, startup, exit, fork, exec, init, halt, poll, strategy, print, _tty,* or *intr*).

When you invoke the *configure* program to modify the system configuration files with the new driver information, you provide the interrupt priority level with the **-l** option and the relevant routine names with the **-a** option.

See Also

configure(ADM), mdevice(F), ''Adding Device Drivers with the Link Kit'' in the *System Administrator's Guide*

Value Added

routines is an extension of AT&T System V provided in Altos UNIX System V.

runacct

run daily accounting

Syntax

/usr/lib/acct/runacct [mmdd [state]]

Description

runacct is the main daily accounting shell procedure. It is normally initiated via *cron*(C). *runacct* processes connect, fee, disk, and process accounting files. It also prepares summary files for *prdaily* or billing purposes. *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*(C)] is sent to **root** and **adm**, and *runacct* terminates. *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 following order:

SETUP Move active accounting files into working files.

WTMPFIX 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, process, 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*.*mmdd*

See Also

acct(ADM), acctcms(ADM), acctcom(C), acctcon(ADM), acctmerg(ADM), acctprc(ADM), acctsh(ADM), cron(C), fwtmp(ADM), mail(C), acct(S), acct(F), utmp(F)

Notes

Normally, it is not a good idea to restart *runacct* in the **SETUP** *state*. Run **SETUP** manually and restart via:

runacct *mmdd* **WTMPFIX**

If *runacct* failed in the **PROCESS** *state*, remove the last **ptacct** file because it will not be complete.

Standards Conformance

runacct is conformant with:

AT&T SVID Issue 2, Select Code 307-127.

sag

system activity graph

Syntax

sag [options]

Description

The *sag* command graphically displays the system activity data stored in a binary data file by a previous *sar*(ADM) run. Any of the *sar* data items may be plotted singly, or in combination; as cross plots, or versus time. Simple arithmetic combinations of data may be specified. The *sag* command invokes *sar* and finds the desired data by string-matching the data column header (run *sar* to see what is available). These *options* are passed through to *sar*:

-s *time* Select data later than *time* in the form *hh[:mm]*. Default is 08:00.

-e *time* Select data up to *time*. Default is 18:00.

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

-f *file* Use *file* as the data source for *sar*. Default is the current daily data file **/usr/adm/sa/sa***dd*.

Other *options*:

-T *term* Produce output suitable for terminal *term*. See *tplot*(ADM) for known terminals. Default for *term* is **$TERM**.

-x *spec* x axis specification with *spec* in the form: "name [op name] ... [lo hi]"

-y *spec* y axis specification with *spec* in the same form as above.

Name is either a string that will match a column header in the *sar* report, with an optional device name in square brackets, e.g., **r+w/s[dsk-1]**, or an integer value. *Op* is + - * or / surrounded by blanks. Up to five names may be specified. Parentheses are not recognized. Contrary to custom, + and - have precedence over * and /. Evaluation is left to right. Thus A / A + B * 100 is evaluated (A/(A+B))*100, and A + B / C + D is (A+B)/(C+D). *Lo* and *hi* are optional numeric scale limits. If unspecified, they are deduced from the data.

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

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

Examples

To see today's CPU utilization:
sag

To see activity over 15 minutes of all disk drives:

```
TS=date +%H:%M
sar -o tempfile 60 15
TE=date +%H:%M
sag -f tempfile -s $TS -e $TE -y "r+w/s[dsk]"
```

Files

/usr/adm/sa/sadd daily data file for day *dd.*

See Also

sar(ADM), tplot(ADM)

sar, sa1, sa2, sadc

system activity report package

Syntax

sar [**-ubdycwaqvmnprDSAC**] [**-o** file] t [n]

sar [**-ubdycwaqvmnprDSAC**] [**-s** time] [**-e** time] [**-i** sec] [**-f** file]

/usr/lib/sa/sadc [t n] [ofile]

/usr/lib/sa/sa1 [t n]

/usr/lib/sa/sa2 [**-ubdycwaqvmnprDSAC**] [**-s** time] [**-e** time] [**-i** sec]

Description

sar, in the first instance, samples cumulative activity counters in the operating system at *n* intervals of *t* seconds, where *t* should be 5 or greater. If the **-o** option is specified, it saves the samples in *file* in binary format. The default value of *n* is 1. In the second instance, with no sampling interval specified, *sar* extracts data from a previously recorded *file*, either the one specified by the **-f** option or, by default, the standard system activity daily data file **/usr/adm/sa/sa***dd* for the current day *dd*. The starting and ending times of the report can be bounded via the **-s** and **-e** *time* arguments of the form *hh*[:*mm*[:*ss*]]. The **-i** option selects records at *sec* second intervals. Otherwise, all intervals found in the data file are reported.

In either case, subsets of data to be printed are specified by option:

-u Report CPU utilization (the default):
%usr, %sys, %wio, %idle - portion of time running in user mode, running in system mode, idle with some process waiting for block I/O, and otherwise idle. When used with **-D**, %sys is split into percent of time servicing requests from remote machines (%sys remote) and all other system time (%sys local).

-b Report buffer activity:
bread/s, bwrit/s - transfers per second of data between system buffers and disk or other block devices;
lread/s, lwrit/s - accesses of system buffers;
%rcache, %wcache - cache hit ratios, i. e., (1-bread/lread) as a percentage;
pread/s, pwrit/s - transfers via raw (physical) device mechanism. When used with **-D**, buffer caching is reported for locally-

mounted remote resources.

-d Report activity for each block device, e. g., disk or tape drive. When data is displayed, the device specification *dsk-* is generally used to represent a disk drive. The device specification used to represent a tape drive is machine dependent. The activity data reported is:
%busy, avque - portion of time device was busy servicing a transfer request, average number of requests outstanding during that time;
r+w/s, blks/s - number of data transfers from or to device, number of bytes transferred in 512-byte units;
avwait, avserv - average time in ms. that transfer requests wait idly on queue, and average time to be serviced (which for disks includes seek, rotational latency, and data transfer times).

-n Report name cache statistics. The activity reported is:
c_hits, cmisses - number of name cache hits and misses;
hit% - the hit ratio as a percentage.

-y Report TTY device activity:
rawch/s, canch/s, outch/s - input character rate, input character rate processed by canon, output character rate;
rcvin/s, xmtin/s, mdmin/s - receive, transmit and modem interrupt rates.

-c Report system calls:
scall/s - system calls of all types;
sread/s, swrit/s, fork/s, exec/s - specific system calls;
rchar/s, wchar/s - characters transferred by read and write system calls. When used with **-D**, the system calls are split into incoming, outgoing, and strictly local calls.

-w Report system swapping and switching activity:
swpin/s, swpot/s, bswin/s, bswot/s - number of transfers and number of 512-byte units transferred for swapins and swapouts (including initial loading of some programs);
pswch/s - process switches.

-a Report use of file access system routines:
iget/s, namei/s, dirblk/s.

-q Report average queue length while occupied, and % of time occupied:
runq-sz, %runocc - run queue of processes in memory and runnable;
swpq-sz, %swpocc - swap queue of processes swapped out but ready to run.

-v Report status of process, inode, file tables:
text-sz, proc-sz, inod-sz, file-sz, lock-sz - entries/size for each table, evaluated once at sampling point;

ov - overflows that occur between sampling points for each table.

-m Report message and semaphore activities:
msg/s, sema/s - primitives per second.

-p Report paging activities:
vflt/s - address translation page faults (valid page not in memory);
pflt/s - page faults from protection errors (illegal access to page) or "copy-on-writes";
pgfil/s - vflt/s satisfied by page-in from file system;
rclm/s - valid pages reclaimed for free list.

-r Report unused memory pages and disk blocks:
freemem - average pages available to user processes;
freeswap - disk blocks available for process swapping.

-D Report Remote File Sharing activity:
When used in combination with **-u**, **-b**, or **-c**, it causes *sar* to produce the remote file sharing version of the corresponding report. **-Du** is assumed when only **-D** is specified.

-S Report server and request queue status:
Average number of Remote File Sharing servers on the system (serv/lo-hi), % of time receive descriptors are on the request queue (request %busy), average number of receive descriptors waiting for service when queue is occupied (request avg lgth), % of time there are idle servers (server %avail), average number of idle servers when idle ones exist (server avg avail).

-A Report all data. Equivalent to **-udqbwcayvmprSDC**.

-C Report Remote File Sharing buffer caching overhead:
snd-inv/s - number of invalidation messages per second sent by your machine as a server.
snd-msg/s - total outgoing RFS messages sent per second.
rcv-inv/s - number of invalidation messages received from the remote server.
rcv-msg/s - total number of incoming RFS messages received per second.
dis-bread/s - number of buffer reads that would be eligible for caching if caching were not turned off. (Indicates the penalty of running uncached.)
blk-inv/s - number of buffers removed from the client cache.

Examples

To see today's CPU activity so far:

sar

To watch CPU activity evolve for 10 minutes and save data:

sar -o temp 60 10

To later review disk and tape activity from that period:

sar -d -f temp

Data Gathering

The operating system contains several 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, interprocess 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, with an interval of *t* seconds between samples, and writes in binary format to *ofile* or to standard output. The sampling interval *t* should be greater than 5 seconds; otherwise, the activity of *sadc* itself may affect the sample. 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/sadate +%d"
```

The shell script *sa1*, a variant of *sadc*, is used to collect and store data in binary file **/usr/adm/sa/sa*dd*** 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*(C)]:

```
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* writes a daily report in file **/usr/adm/sa/sar*dd***. The **/usr/spool/cron/crontabs/sys** entry:

```
5 18 * * 1-5 /usr/lib/sa/sa2 -s 8:00 -e 18:01 -i 1200 -A
```

will report important activities hourly during the working day.

The structure of the binary daily data file is:

```
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 */
      struct rcinfo rc;   /* Client cache info defined in sys/sysinfo.h */
      struct bpbinfo bi;  /* Co-processor info defined in sys/sysinfo.h */
      int bpb_utilize     /* Co-processor utilize flag */
      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  */
      int                 cachehits;/* number of name cache hits */
      int                 cachemisses;/* number of name cache misses */
#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/sa*dd*	daily data file
/usr/adm/sa/sar*dd*	daily report file
/tmp/sa.*adrfl*	address file

Notes

Output files created with this version of *sar* cannot be interpreted by earlier versions. However, this version interprets older output files correctly.

See Also

cron(C), sag(ADM), timex(ADM)

Standards Conformance

sa1, *sa2*, *sadc* and *sar* are conformant with:
AT&T SVID Issue 2, Select Code 307-127.

schedule

database for automated system backups

Description

The *schedule* database is used in conjunction with *fsphoto*(ADM) to partially automate system-wide backups. For each filesystem to be backed-up, a cyclical schedule of *xbackup*(ADM) or *cpio*(C) levels is specified. (*fsphoto* uses *cpio*(C) or *xbackup*(ADM), for XENIX or for UNIX filesystems, respectively.)

This cyclical schedule (or *cycle*) is a list of dump levels to perform (including no dump at all) and a pointer to the last-used element of that list. The pointer is advanced to the next element of the list on a regular basis (each time *fsphoto* is run, usually once per day), starting over at the beginning each time it falls off the end. It is advanced, however, only on success - the desired dump must have been successful.

Each entry in the file is on a separate line. Blank and comment lines (beginning with "#") may be placed anywhere. Several keywords are recognized:

site *sitename*
Sitename is passed to *fsave* as a description to place on each tape label. Usually, *sitename* is the name of the company or a building number.

media *drive* **k** *sizes...* [*format*]
Device *drive* is a floppy capable of handling volumes with any of the listed *sizes* (in kilobytes). If specified, *format* is the UNIX command used to format the described floppies.

media *drive* **d** *density sizes...* [*format*]
Device *drive* is a *density* BPI magtape capable of handling tapes of any of the indicated *sizes* (in feet). Like floppies, *format* is the optional UNIX command used to format the described tape.

[0-9] *size savetime importance marker*
Description of each dump level, as described in *fsave*(ADM). The defaults are:

Level	Size	Savetime	Importance	Marker
0	-	"1 year"	critical	none
1	-	"3 months"	necessary	none
2...7	-	"1 month"	important	none
8	-	"2 weeks"	useful	none
9	-	"1 week"	precautionary	none

All four fields must be specified. A *size* of - means to use the first size listed in the appropriate **media** *sizes* list.

Keywords should be placed before any filesystem dump schedules. A filesystem dump schedule is of the form:

/dev/rfilesys *cycle*
The filesystem resident on device **/dev/filesys** is to be backed-up according to *cycle*, which is a space-separated list of dump levels (the digits **0** to **9**, passed to *dump*), or the letter **x**, meaning no dump should occur.

A dump *cycle* must have at least one member, but it may be of any length. Different filesystems may have *cycles* of different lengths.

Here is a sample *schedule* file:

```
# SYSTEM BACKUP SCHEDULE
site mymachine

# Media Entries
# 96 tpi 1.2 MB floppy 0
# media /dev/rfd096ds15 k 1200 format /dev/rfd096ds15
# 96 tpi 1.2 MB floppy 1
# media /dev/rfd196ds15 k 1200 format /dev/rfd196ds15
# Cartridge tape 0
# media /dev/rct0 d 20000 300 450 600 tape erase
# 9-track tape drive
# media /dev/rmt0 d 1600 2400 1200 600
# Backup Descriptor Table
#   Backup   Vol.   Save for     Vitality        Label
#   level    size   how long     (importance)    marker
    0        -      "1 year"     critical        "a red sticker"
    1        -      "4 months"   necessary       "a yellow sticker"
    8        -      "3 weeks"    useful          "a blue sticker"
    9        -      "1 week"     precautionary   none
# Schedule Table
#                1 2 3 4 5   6 7 8 9 0   1 2 3 4 5   6 7 8 9 0
# Filesystem     MTWTF       MTWTF       MTWTF       MTWTF       Method
/dev/rroot       0 x 9 x 9   8 x 9 x 9   1 x 9 x 9   8 x 9 x 9   cpio
/dev/ru          9 0 9 9 9   9 8 9 9 9   9 1 9 9 9   9 8 9 9 9   cpio
```

In the example above, filesystem **/dev/rroot** is dumped using a level 0 dump the first time *fsphoto* is run (on a Monday), and if that dump is successful, the next (second) time it runs (Tuesday), no dump is performed. If doing nothing is successful, the third time run (Wednesday) a level 9 dump occurs. If that dump succeeds, no dump occurs the fourth time (Thursday), but the fifth time *fsphoto* is run (Friday), a level 9 dump is made.

Each time a successful dump at the specified level happens, the pointer advances so that the next run of *fsphoto* (on the next weekday) will do the next dump scheduled for that filesystem. If however, a dump fails (or is interrupted or postponed by the operator) the pointer is not advanced; hence, the next time *fsphoto* is attempted, the same level dump will again be tried so the sequence will not be broken (but the timing may be off).

Continuing the example, the nineteenth time *fsphoto* runs, a level 9 dump of /dev/rroot is done, no dump is performed the next (twentieth) time, but the twenty-first time (Monday of every fifth week) the cycle starts over again at the beginning with a level 0 dump.

The larger and more rapidly changing filesystems /dev/ru is dumped more frequently (each time *fsphoto* is run - once a day - instead of every other time), and the levels used are staggered to prevent having to perform two full-scale dumps (like levels 0 or 1) of the large filesystems on the same day. The backup cycle period is also shorter, two weeks instead of four.

The *Method* field defines the backup utility to be used. *cpio* works for both XENIX and UNIX filesystems, but *xbackup* works only on XENIX filesystems.

See Also

fsphoto(ADM), fsave(ADM), xbackup(ADM)

Notes

Keywords and filesystem names must not be preceded by any spaces or tabs.

It is not necessary to specify the name of the "raw" (/dev/*r**) device for each filesystem, but the backups are faster if this is done.

Value Added

schedule is an extension of AT&T System V provided in Altos UNIX System V.

scsinfo

display current SCSI device information

Syntax

scsinfo [**-q**] [[**-t** | [**-n** | **-p** *devname*] | [**-s** | **-i** *index#*]]

Description

scsinfo displays information about all the current SCSI hard disks and tape drives, or one in specific. For each device, it displays the following:

- device name
- logical SCSI index (not including SCSI controller)
- major and minor numbers (hard disks only)
- slot number of the card it resides in
- host adapter driver prefix
- host adapter number (channel number)
- SCSI ID
- SCSI Logical Unit Number (LUN)

Typing **scsinfo** with no options displays information for all SCSI hard disks currently loaded on the system. Type **scsinfo -t** to display information on all SCSI tape drives.

You may also request information for specific devices only. Identify such devices with either their device name (**-p** for tapes, **-n** for disks), or SCSI index number (**-s** for disks, **-i** for tapes).

Options

Options to *scsinfo* are:

-q Enable quiet mode. Displays hard disk information (as with default use), but with no headers and no prompting for page breaks.

-t Display information for all tape drives currently installed.

-n *devname*
Display information for the specified hard disk, as identified by its device name, *devname*.

-p *devname*
Display information for the specified tape drive, as identified by its device name, *devname*.

-s *index#*
Display information for the specified hard disk, as identified by its logical SCSI index number, *index#*. Logical index numbers for SCSI disks range from 0 to 51.

-i *index#*
Display information for the specified tape drive, as identified by its logical SCSI index number, *index#*. Logical index numbers for SCSI tapes range from 0 to 7.

Notes

For more infomation on SCSI hard disk device numbering, see the table found in *divvy* (ADM).

Value Added

scsinfo is an extension to AT&T System V provided in Altos UNIX System V.

setclock

sets the system real-time (time of day) clock

Syntax

setclock [time]

Description

The **setclock** file sets the battery-powered, real-time time of day clock to the given *time*. If *time* is not given, the current contents of the battery-powered clock are displayed. The *time* must be a combination of digits with the form:

MMddhhmmyy

where *MM* is the month, *dd* is the day, *hh* is the hour, *mm* is the minute, and *yy* is the last two digits of the year. If *yy* is not given, it is taken from the current system time. For example, the command:

082615035

sets the time of day clock to 15:03 on August 26, 1985.

Files

/etc/setclock

See Also

clock(F)

Notes

Not all computers have battery-powered real-time time of day clocks. Refer to your computer's hardware reference manual.

Value Added

setclock is an extension of AT&T System V provided in Altos UNIX System V.

setmnt

establishes /etc/mnttab table

Syntax

/etc/setmnt

Description

setmnt creates the **/etc/mnttab** table (see *mnttab*(F)), which is needed for both the *mount*(ADM) and *umount*(ADM) commands. *setmnt* reads the standard input and creates a *mnttab* entry for each line. Input lines have the format:

filesys node

where *filesys* is the name of the file system's *special file* (e.g., "hd0") and *node* is the root name of that file system. Thus *filesys* and *node* become the first two strings in the *mnttab*(F) entry.

Files

/etc/mnttab

See Also

mnttab(F)

Notes

If *filesys* or *node* are longer than 128 characters, errors can occur.

setmnt silently enforces an upper limit on the maximum number of *mnttab* entries.

setmnt is normally invoked by the **/etc/rc2** scripts when the system boots up.

settime

changes the access and modification dates of files

Syntax

settime mmddhhmm [yy] [**-f** fname] name ...

Description

Sets the access and modification dates for one or more files. The dates are set to the specified date, or to the access and modification dates of the file specified via **-f**. Exactly one of these methods must be used to specify the new date(s). The first *mm* is the month number; *dd* is the day number in the month; *hh* is the hour number (24 hour system); the second *mm* is the minute number; *yy* is the last two digits of the year and is optional. For example:

settime 1008004583 ralph pete

sets the access and modification dates of files *ralph* and *pete* to Oct 8, 12:45 AM, 1983. Another example:

settime -f ralph john

This sets the access and modification dates of the file *john* to those of the file *ralph*.

Notes

Use of *touch* in place of *settime* is encouraged.

shutdown

terminates all processing

Syntax

/etc/shutdown [-y] [-g[hh:]mm] [-i[0156sS]] [-f"mesg"] [-fFILE] [su]

Description

The primary function of *shutdown* is to terminate all currently running processes in an orderly and cautious manner. *shutdown* goes through the following steps:

1. All users logged on the system are notified to log off the system by a broadcasted message.

2. **/etc/init** is called to perform the the actual shutdown.

Only the super-user can execute the *shutdown* command.

The options are as follows:

-y Runs the command silently. If this option is not specified, *shutdown* will prompt for confirmation to shut down the system.

-g[hh:]mm Specifies the number of hours and minutes before shutdown (maximum: 72 hours). 60 seconds is the default. (To shut down the system immediately without a grace period, use **/etc/haltsys** or **/etc/reboot**).

-i[0156sS] Specifies the init level to bring the system to (see *init*(M)). By default, the system is brought to level 0.

-fmesg *mesg* is a message enclosed in double quotes ("") to be sent to all terminals warning of the imminent shutdown during the grace period.

-fFILE Similar to the -fmesg option, but FILE is the pathname for a file containing the message.

The optional *su* argument lets the user go single-user, without completely shutting down the system (this option is identical to **-i1** and is present for backwards compatibility with XENIX). Broadcast messages, whether default or defined, are displayed at regular intervals during the grace period. The closer the shutdown time, the more frequent the message:

Time left until shutdown	Frequency of message
Greater than 1 hour	Every hour
Greater than 15 minutes	Every 15 minutes
Less than 15 minutes	Every minute

In general, if no options are specified, *shutdown* behaves as follows:

1. Prompt for confirmation
2. 60-second grace period
3. Bring the system to init level 0
4. Broadcast default message prior to shutdown.

See Also

wall(ADM), boot(HW)

Diagnostics

The most common error diagnostic that will occur is *device busy*. This diagnostic appears when a particular file system could not be unmounted. See *umount*(ADM).

Notes

Once *shutdown* has been invoked, it must be allowed to run to completion and must *not* be interrupted by pressing BREAK or DEL.

shutdown does not work when executed from within a shell layer.

shutdown locks the hard disk heads.

shutdown was developed at the University of California, Berkeley, and is used with permission.

strace

prints STREAMS trace messages

Syntax

strace [mid sid level] ...

Description

strace without arguments writes all STREAMS event trace messages from all drivers and modules to its standard output. These messages are obtained from the STREAMS log driver [*log*(M)]. If arguments are provided they must be in triplets of the form *mid, sid, level,* where *mid* is a STREAMS module id number, *sid* is a sub-id number, and *level* is a tracing priority level. Each triplet indicates that tracing messages are to be received from the given module/driver, sub-id (usually indicating minor device), and priority level equal to or less than the given level. The token *all* may be used for any member to indicate no restriction for that attribute.

The format of each trace message output is:

<seq> <time> <ticks> <level> <flags> <mid> <sid> <text>

<seq>	trace sequence number
<time>	time of message in hh:mm:ss
<ticks>	time of message in machine ticks since boot
<level>	tracing priority level
<flags>	E : message is also in the error log F : indicates a fatal error N : mail was sent to the system administrator
<mid>	module id number of source
<sid>	sub-id number of source
<text>	formatted text of the trace message

Once initiated, *strace* will continue to execute until terminated by the user.

Examples

Output all trace messages from the module or driver whose module id is 41:

strace 41 all all

Output those trace messages from driver/module id 41 with sub-ids 0, 1, or 2:

strace 41 0 1 41 1 1 41 2 0

Messages from sub-ids 0 and 1 must have a tracing level less than or equal to 1. Those from sub-id 2 must have a tracing level of 0.

See Also

log(M), *STREAMS Programmer's Guide*

Diagnostics

Due to performance considerations, only one *strace* process is permitted to open the STREAMS log driver at a time. The log driver has a list of the triplets specified in the command invocation, and compares each potential trace message against this list to decide if it should be formatted and sent up to the *strace* process. Hence, long lists of triplets will have a greater impact on overall STREAMS performance. Running *strace* will have the most impact on the timing of the modules and drivers generating the trace messages that are sent to the *strace* process. If trace messages are generated faster than the *strace* process can handle them, then some of the messages will be lost. This last case can be determined by examining the sequence numbers on the trace messages output.

strclean

STREAMS error logger cleanup program

Syntax

strclean [-d logdir] [**-a** age]

Description

strclean is used to clean up the STREAMS error logger directory on a regular basis (for example, by using *cron*(C)). By default, all files with names matching **error.*** in **/usr/adm/streams** that have not been modified in the last 3 days are removed. A directory other than **/usr/adm/streams** can be specified using the **-d** option. The maximum age in days for a log file can be changed using the **-a** option.

Example

strclean -d /usr/adm/streams -a 3

has the same result as running *strclean* with no arguments.

Notes

strclean is typically run from *cron*(C) on a daily or weekly basis.

Files

/usr/adm/streams/error.*

See Also

cron(C), strerr(ADM), *STREAMS Programmer's Guide*

strerr

STREAMS error logger daemon

Syntax

strerr

Description

strerr receives error log messages from the STREAMS log driver [*log*(M)] and appends them to a log file. The error log files produced reside in the directory **/usr/adm/streams**, and are named **error.***mm-dd*, where *mm* is the month and *dd* is the day of the messages contained in each log file.

The format of an error log message is:

<seq> <time> <ticks> <flags> <mid> <sid> <text>

<seq>	error sequence number
<time>	time of message in hh:mm:ss
<ticks>	time of message in machine ticks since boot priority level
<flags>	T : the message was also sent to a tracing process F : indicates a fatal error N : send mail to the system administrator
<mid>	module id number of source
<sid>	sub-id number of source
<text>	formatted text of the error message

Messages that appear in the error log are intended to report exceptional conditions that require the attention of the system administrator. Those messages which indicate the total failure of a STREAMS driver or module should have the F flag set. Those messages requiring the immediate attention of the administrator will have the N flag set, which causes the error logger to send the message to the system administrator via *mail*(C). The priority level usually has no meaning in the error log but will have meaning if the message is also sent to a tracer process.

Once initiated, *strerr* will continue to execute until terminated by the user. Commonly, *strerr* would be executed asynchronously.

Notes

Only one *strerr* process at a time is permitted to open the STREAMS log driver.

If a module or driver is generating a large number of error messages, running the error logger will cause a degradation in STREAMS performance. If a large burst of messages are generated in a short time, the log driver may not be able to deliver some of the messages. This situation is indicated by gaps in the sequence numbering of the messages in the log files.

Files

/usr/adm/streams/error.*mm-dd*

See Also

log(M), *STREAMS Programmer's Guide*

strmcfg

STREAMS configuration utility for networking products

Syntax

/altos/bin/strmcfg [**-h**] [**-d** directoryname] [**-f** filename]

Description

strmcfg is a STREAMS configuration utility that calculates optimum values for selected STREAMS parameters based on the various conditions under which your networking products are running. You may also use *strmcfg* to calculate STREAMS parameters for a selected product or group of products.

When invoked, *strmcfg* reads each data file (one file per product) in the default directory **/usr/altos/tuning/streams/data** and passes the configuration requests (or *questions*) contained in each file to the user. *strmcfg* uses these responses (along with other data defined in the file) to calculate optimal STREAMS parameter values. After all questions in every file are asked, *strmcfg* writes the resulting parameter values to an output file and then exits. The default output file is **/usr/altos/tuning/streams/results.** The parameter values in this file can be implemented in the kernel with the *idtune*(ADM) command, or you can use the *strmtune*(ADM) utility to configure these values in your kernel. In an adjoining directory, **./config**, there are data files for each product. These data files contain information about the parameter values set by each product. The default direcotry is **/usr/altos/tuning/streams/config**. If you use *strmtune*(ADM) to configure the kernel using the results file, product files with the extension **.cfg** are also saved in this directory.

The **results** file is overwritten each time *strmcfg* is invoked, so if you need the existing results file, move it before invoking *strmcfg* (or use the **-f** option, described below). The product files are also rewritten.

Here is a list of the configurable STREAMS parameters:

NBLK4	NBLK1024	NNFSRNODE
NBLK16	NBLK2048	NQUEUES
NBLK64	NBLK4096	NS5INODE
NBLK128	NFILE	NUMTIM
NBLK256	NINODE	NUMTRW

NBLK512 NMOUNT NSTREAM

Options

-h Print a brief help message about *strmcfg*.

-d *directoryname* Search the directory *directoryname* for the data files. If not specified, the default directory **/usr/altos/tuning/streams/data** is searched.

-f *filename* Write the resulting STREAM parameter values in the file *filename*. If not specified, the default output file **/usr/altos/tuning/streams/results** is used. The product-specific configuration files will be stored in an adjoining directory, **./config**.

-p Print information about your current configuration.

Data File Format

This section describes how to modify a data file, or create a new one.

The data file contains the following special strings:

PRODUCT:
QUESTION:
ANSWER:
PARAMETERS:
n:
#

Every line in the data file must begin with one of these strings. There cannot be any white spaces before the special strings/characters; they must occur at the start of each line in column one. Also, data lines cannot have any carriage return or new-line characters in them, except at the end of the line.

Each string and its use are described below:

PRODUCT: The product name should follow this string, and should be terminated with a colon (:).

QUESTION: This string identifies a configuration query, which must be terminated with a colon (:).

ANSWER: All legal answers to the preceding question follow the ANSWER: string. The answers are separated by a comma (,), and the entire line must be terminated by a comma too.

Each answer is used later in the file to identify the exact parameter values it changes. See *n*: below.

PARAMETERS: The line prefaced by the PARAMETRS: string should contain all the STREAMS parameters that are affected by the preceding question. As with the ANSWERS: string, the parameters must be separated by a comma (,), and the line terminated with one too.

n: For every valid answer *n* defined in the ANSWER: string, there should be a line starting with *n*: that specifies how the parameter values are changed if this answer is selected. The *n*: line contains pairs of values, one pair for each parameter listed in the PARAMETRS: string, in the order these parameters are listed. The first number in the pair defines the minimum value of the corresponding parameter when the answer *n* is selected. The second number defines the "step" value by which the current parameter value is incremented (if already at or above the minimum value).

Each number in this string is separated by a comma (,), and the entire line is terminated with a comma.

Any line starting with the pound character is ignored. Use this character for comment lines in the data file.

An example data file is listed below.

```
# streams tuning script for TCP/IP
#
# the name of the product for which this file is used
#
PRODUCT: Altos TCP/IP:
QUESTION: Indicate maximum circuits :
#
# the only valid answers are 8, 16, 32, and 64
#
ANSWER:8,16,32,64,
#
# the parameters NBLK16, NBLK64, NBLK128, NBLK1024
# are modified depending upon the answer supplied
#
PARAMETERS:NBLK16,NBLK64,NBLK128,NBLK1024,
#
# if '8' is selected, the various parameters are changed by the
# (minimum,step) pairs (50,10), (150,11), etc.
# this means that if the NBLK16 value is not at least '50' it will be
```

```
# set to that, otherwise the current value will be incremented by '1C
#
8:50,10,150,11,150,10,290,25,
16:60,10,120,30,180,10,290,25,
32:80,10,160,50,240,10,290,25,
64:100,10,200,70,300,10,290,25,
#
QUESTION: Question Two ? :
ANSWER:8,16,32,64,
PARAMETERS:NBLK16,NBLK64,NBLK128,NBLK1024,
8:50,10,150,11,150,10,290,25,
16:60,10,120,30,180,10,290,25,
32:80,10,160,50,240,10,290,25,
64:100,10,200,70,300,10,290,25,
```

Files

/usr/altos/tuning/streams/data
/usr/altos/tuning/streams/results
/usr/altos/tuning/streams/config

See Also

idtune(ADM)

Value Added

strmcfg is an extension to AT&T System V provided in Altos UNIX System V.

strmtune

STREAMS configuration interface for networking products

Syntax

/usr/altos/bin/strmtune

Description

This utility helps you configure your kernel for specific communication products. It lets you run the *strmcfg*(ADM) utility, tune your kernel STREAM parameters in accordance with output of *strmcfg*, decrease the STREAMS parameters you have configured, and finally view your current STREAMS parameter configuration.

Options

1. Run strmcfg with default options

   This option runs the *strmcfg* utility with the default options, i.e. the data files will be read from **/usr/altos/tuning/streams/data**, the results file will be **/usr/altos/tuning/streams/results**, and finally the product-specific files will be stored in the directory **/usr/altos/tuning/streams/config.**

2. Tune kernel with new parameters

   This option will configure your kernel in accordance with the results produced by option 1. After all the parameters have been adjusted, you will be asked if you want to relink the kernel. The new configuration will not take effect until you have rebuilt the kernel and rebooted the system.

3. Remove added parameters from kernel

   This option will decrease the kernel parameters you have added using option 2. You will be asked for each product configured, whether or not you want to remove the paramters. It will then adjust your kernel parameters accordingly. After all the parameters have been adjusted, you will be asked if you want to relink the kernel. The new configuration will not be in effect until you have rebuilt the kernel and rebooted the system.

4. Display Streams Parameters

   This option displays your current streams configuration. You should use this option frequently to examine how your changes are affecting your kernel parameters.

Output from Display option

This section will explain how you can interpret the information displayed by option 4. In all examples only a part of the STREAMS parameters will be shown. Option 4 will always show you the configuration of all of the STREAMS paramters.

When you first invoke *strmtune*, the kernel parameters will not be specifically tuned for any communications products. The output of option 4 will look something like this:

```
Current STREAMS Configuration:

Kernel      Current    Communications   Parameter     Proposed new  Proposed new
STREAM      Value in   Product(s)       Increase      Increase      Value for
Parameter:  Kernel:    Configured:      by product:   by product:   parameter:

NSTREAM       192       (none)
NBLK4         256       (none)
NBLK4096        0       (none)
NFILE         512       (none)
```

This output shows you the current value of the STREAMS parameters (column 2). It also shows you that these values have not been affected by configuring any communications packages (column 3).

If you then run option 1 from the menu, and configure for Altos TCP/IP and Altos NFS.

```
Current STREAMS Configuration:

Kernel      Current    Communications   Parameter     Proposed new  Proposed new
STREAM      Value in   Product(s)       Increase      Increase      Value for
Parameter:  Kernel:    Configured:      by product:   by product:   parameter:

NSTREAM       192      Altos TCP/IP          -             10           234
                       Altos NFS             -             32
NBLK4         256      Altos TCP/IP          -             60           316
NBLK4096        0      Altos TCP/IP          -              2             2
NFILE         512       (none)
```

This output shows you the same current value of the STREAMS parameters (column 2) as before. Column 3 now lists the packages that we have configured, and which affects this particular parameter. For example, NSTREAM is affected by Altos TCP/IP and Altos NFS, while NBLK4 is only affected by Altos TCP/IP. Column 5 shows you how much a particular package will increase a parameter, and column

6 shows you what the parameter wili be set to in the kernel if you choose to tune your kernel using this configuration (using option 2 from the main menu). In this case, NSTREAM will increase from 192 to 234, because TCP/IP will add 10 and NFS will add 32.

After you have examined and approved your changes you can go back to the main meny and choose option 2. If you display the stream configuration afterwards, it will look something like this:

Current STREAMS Configuration:

Kernel STREAM Parameter:	Current Value in Kernel:	Communications Product(s) Configured:	Parameter Increase by product:	Proposed new Increase by product:	Proposed new Value for parameter:
NSTREAM	234	Altos TCP/IP	10	-	-
		Altos NFS	32	-	
NBLK4	316	Altos TCP/IP	60	-	-
NBLK4096	2	Altos TCP/IP	2	-	-
NFILE	512	(none)			

This shows you that the value of the STREAMS parameters in the kernel has changed, it also shows you to which communications packages you can contribute this change. If you choose to remove the added parameters (using option 3 from the menu) for TCP/IP, e.g. the NSTREAM parameter would decrease by 10 to 224.

Current STREAMS Configuration:

Kernel STREAM Parameter:	Current Value in Kernel:	Communications Product(s) Configured:	Parameter Increase by product:	Proposed new Increase by product:	Proposed new Value for parameter:
NSTREAM	234	Altos TCP/IP	10	10	266
		Altos DOS Server	-	32	
		Altos NFS	32	-	
NBLK4	316	Altos TCP/IP	60	60	346
		Altos DOS Server	-	30	
NBLK4096	2	Altos TCP/IP	2	4	4
NFILE	512	(none)			

This is an example of how the output will look if you run the strmcfg utility (option 1) after you have already configured the kernel once. In this example configurations for DOS Server is added, and TCP/IP has been changed from 8 to 16 file transfer users. Column 2 shows you what the current kernel values are. Column 4 shows you what the current kernel values are per product. In column 5 you can see the new values for the products you have added or reconfigured. A dash '-' in this column means that you have choosen not to change this product. A new product will have its configurations added to the kernel values. An updated product will have the new configuration replacing the old values.

Files

/usr/altos/tuning/streams/data
/usr/altos/tuning/streams/results
/usr/altos/tuning/streams/config

See Also

strmcfg(ADM)

Value Added

strmtune is an extension to AT&T System V provided in Altos UNIX System V.

submit

MMDF mail enqueuer

Syntax

submit **[-L...*V...*Wbcdf...*g...*hi...*jk...*lmnqrstuvwxyz]**

Description

All mail is entered into the MMDF mail transport environment through the *submit* program. This document is intended to provide the specific information needed to control *submit*. While it can be called directly from a user's terminal, access to *submit* is most conveniently done through a program such as *mail*(C). It also will be useful to read replies(M), which describes reply values.

Basic Modes

submit permits considerable flexibility with respect to batching multiple submissions, response and error handling, and address source specification.

Multiple Submissions

1. Terminate after one submission, such as is done by the mail command, or

2. Permit multiple message submissions, as is done by the SMTP channel and the MMDF telephone *slave*.

The first mode is specified by passing any initialization information in the submit invocation line (i.e., the *exec*(S) call). In the second mode, the initialization information is given as the first input line, for each submission. The format of this information is the same for both modes.

Response & Error Handling

1. Accept input until error or end of message, but terminate on any error, or

2. Notify result for each **segment** and continue.

Response mode #1 is mandatory with Multiple mode #1. Response mode #2 is called *protocol mode*. During it, each address produces a status reply and the message text produces a reply. The domain of the term **segment** depends on the error. Simple addressing errors cause rejection only of the erroneous address. Other errors may cause rejection of the entire message, but permit submission of following messages.

Addresses

1. Extracted from components of the message text,
2. Explicit list given, ahead of message text, or
3. Both of the above (extracted and explicit addresses)

The first mode is common when mode #1 (non-protocol) is also in force for the Interaction and the Verification option. The second mode is commonly in force when the second modes apply for the other options (protocol mode).

Initialization

A message's initialization information is specified through a single string, passed either in the process-invocation argument list or in the first line of *submit* input. Hence, the string may be terminated either by a null or newline. Spaces and tabs in the line are ignored, unless part of a literal. Specification is only required for non-defaults.

	Option	Value	Literal
1.	Relay source for the "Via" or "Received" field	a. none	(*default*)
		b. source channel	i...*
		c. source host	h...*
2.	From/Sender authentication	a. reject on error	(*default*)
		b. trust	t
		c. no trust (disclaim)	u
3.	Source-Info Field	a. not included	(*default*)
		b. disclaim author	u
		c. user text	f...*
4.	Address list source		
		a. explicit list	(*default*)
		b. extract from components	x...*
		c. both (extract and	g...*

	explicit)	
5. Address verification		
	a. abort on invalid	*(default)*
	b. report on each address	v
6. Delivery destination		
	a. mailbox	m *(default)*
	b. user's tty	y
	c. mailbox and tty	b
7. Delivery attempt (combinable)	a. leave for daemon	*(default)*
	b. deliver local now	l
	c. deliver netmail now	n
8. Observation of immediate attempts	a. none	*(default)*
	b. user will watch	w
9. Return address	a. send to submittor	r
	b. send to ''Sender:''	s
	c. do not return	q
	d. as specified	*(next line)*
10. Returned mail contents	a. entire original	*(default)*
	b. citation only	c
11. Warnings	a. send warnings	*(default)*
	b. do not send warnings	z
12. Delay channel usage	a. enable delay channel	*(default)*
	b. don't use delay	d
13. Delay channel indicator	a. not delay channel	*(default)*
	b. delay channel	j
14. Nameserver timeouts	a. short timeouts	*(default)*
	b. as specified	k...*
15. Submission tracing	a. not shown	*(default)*
	b. watch submission	W
16. Logging file	a. as per msglog	*(default)*
	b. as specified	L...*
17. Logging level	a. as per msglog	*(default)*
	b. as specified	V...*

Comments

General

Literals shown as characters, followed by an ellipsis, followed by an asterisk (e.g. x...*), represent a string. The first character specifies the nature of the setting. The value for the setting is placed between that character and the asterisk. The value may be any string not containing an asterisk, null, or newline. The values for settings **x** and **g** are comma-separated lists of strings. These strings may not contain asterisks, nulls, newlines, or commas.

Specific

1. Relaying
   This is used when the calling program is interfacing with another distribution system, effecting relaying. The literal after the **i** specifies the channel the message is coming from. The **h** may be used, in conjunction with **i**, to specify the source host. The literal is the name of the host.

2. Authentication
   Normally, the message must correctly identify its sender. Anyone may send "anonymous" (unsigned) mail, but they must use the **u** setting which bypasses authentication. However, it also causes MMDF to include, in the Source-Info: component, a statement noting the absence of authentication. Only root or relays may use the **t** setting, which bypasses authentication and does not add a disclaimer. Others requesting it get **u** treatment.

3. Source-Info
   In addition to the action explained above, Source-Info: can directly receive text, from the user, through the **f** setting. The value string is replicated on a separate line in the field.

4. Address lists
   An explicit list has one address per line. When **x** or **g** are specified, they list the names of message components, such as "To:" and "CC:", which are to be searched for addresses.

5. Verification
   Normally, any illegal address will cause the entire message to be rejected. In **v** (verify) mode, the acceptability of each message is reported and encountering an illegal address does not abort submission.

6. Delivery type
   Mail may be delivered to a recipient's mailbox (file), online terminal (if the recipient is logged in), or a combination of the two. There is no default. For each message, its delivery mode must be

specified.

7. Attempt
   An immediate attempt causes a special *deliver* process to be forked and it will attempt to process the indicated mail immediately. (The **n** setting does not allow more granularity, for historical reasons.) Otherwise, the system's background daemon will get to it eventually. The daemon also handles mail that initially could not be delivered/relayed. A channel's descriptor structure (in *chan.c* or the runtime tailor file) specifies a channel as being Active, Passive, or Background. Only the first is processed by any request for immediate delivery. The second indicates a Post Office Box-style channel. The third limits the channel to processing by the background *deliver* daemon, which may be necessary for restricting access to special channels, such as dial-out telephones.

8. Observation
   If an immediate attempt is requested, the user may elect to watch its progress. *Deliver* and its children will report assorted aspects of their activity. If a quiet attempt is requested, *submit* returns as soon as submission is completed. That is, a quiet attempt is performed detached.

9. Return address
   If the invoker of *submit* is not to receive return mail (e.g., notification of delivery failure) then the next input line (the first, if settings are specified in the *exec*(S) call), contains an address that should receive the notification. It is not validated. If either the **r** or the **s** switch is given, *submit* will not read a line for the return address. If no return mail should be sent, the return address line should be empty (i.e., consist of a newline, only.) If the **q** switch is given, a return address is read from the next line of input but the local system will not return mail if delivery problems are encountered. The return address given may be used by other systems (if there are mail relays between the local system and the recipient).

10. Return contents
    Normally, a copy of the entire message is sent with a delivery-failure notice. Using the **c** switch causes a citation, comprising the message header and first three lines of non-blank lines of the body, to be sent. If more than 12 addresses are specified, for a message, citation-only is automatically set. In addition, no warning message will be sent for addresses which take a long time to process (a site dependent value); the final failure notice will always be sent, if there are addresses that are never fully processed.

11. Warnings
    Normally MMDF will send a non-delivery warning if a message has been undelivered after a small period (typically 12 to 72 hours, depending on the site). Deliver attempts continue until a timeout period is reached. This is typically after 3 to 10 days, depending

on the site.

12. Disable delay channel
    The delay channel is used to process mail submissions that could not queued because necessary nameserver information was unavailable and therefore an authoritative decision on the validity of the address was not possible. If the **d** option is specified, use of the delay channel is prohibited. If the nameserver fails, a error is returned, rather than a conditional OK.

13. Delay channel indicator
    This option is intended only to be used by the delay channel itself to indicate to submit that the invoking process IS the delay channel. This option implies the **d** option above.

14. Nameserver timeouts
    By default, MMDF uses a short timeout algorithm. This is suitable for user interface programs which don't want to wait a long time for dead nameservers. The **k** option allows a different timeout to be set. The value given is the number of seconds to wait for the nameserver lookup to complete.

15. Submission tracing
    The **W** option causes submit to print a detailed description of its activities on file descriptor 2. It will indicate, for each addressee, the channel and addresses queued. This can generate a great deal of output if a mailing list is encountered, so it should be used with caution.

16. Logging file
    The **L** option allows the specification of an alternate logging file at runtime. The string following the **L** should be the name of the logfile to be used. It can be terminated by a * or the end of the arguments. This option is only available to the Superuser or MMDF.

17. Logging level
    The **V** option allows the setting of the logging level at runtime. The string following the **V** should be one of the valid MMDF logging level strings such as FTR or BST. It can be terminated by a * or the end of the arguments. This option is only available to the Superuser or MMDF.

Input Stream

The following augmented BNF characterizes submit's input (file descriptor zero) format:

stream:	*(init-seq '\n' msg-info null) [null]
init-seq:	*{ switches listed above }
msg-info:	[ret-addr] '\n' [addr-seq '!' '\n'] { rfc822-format message }
ret-addr:	{ rfc822-format (return) address }
addr-seq:	*{ rfc822-address }

Address Format

Addresses are expected to conform to the ARPANET mail standard known as RFC-822, available from the Network Information Center at SRI International. *submit* (and MMDF in general) also continues to support RFC-733 style mail for compatibility with earlier mail systems.

In addition to those in RFC-822, the following address delimiters are recognized within the local part of addresses (in order of precedence):

@
%
!
.

The "!" delimiter is interpreted as "host!user" while the others are interpreted as "user?host". For example, the address "a.b!user%c@localhost" would be queued for "a.b!user@c". The address "a.b!user@localhost" would be queued for "user@a.b". The address "user.a@localhost" would be queued for "user@a". Note that recognition of the "." delimiter is a site-selectable option.

Also, addresses may be indirectly referenced, through a file specification of the form:

"<filename" or ":include:filename"

where the angle-bracket must be the first non-blank character of the specification (to distinguish it from the "<...>" usage, above).

Addresses in the file may be separated by commas or newlines.

Example Interactions

Phases involve Invocation (Invoke), data sent into *submit* via its file descriptor zero (To), data returned from *submit* via its file descriptor one (From), iteration back to the specified phase (Loop), and process exit value (Exit).

1. Simple, single-message, as with the *v6mail* command:

a. Invoke:	Parameters, ‘‘-mlrxto,cc*’’, indicate that the message is to be sent to recipients’ mailboxes, local mail should be sent immediately, return mail goes to the submittor, and addresses are to be extracted from the ‘‘To:’’ and ‘‘cc:‘‘ components.
b. To:	The entire message
c. From:	Error messages
d. Exit:	Process return value, in wait(&val), taken from *mmdf.h,* indicating submission status.

2. Standard, multi-message protocol:

a. Invoke:	No parameters
b. To:	Initialization information line. A typical user program might have "mlrv", indicating the message is to be sent to mailboxes, local mail sent immediately, return mail goes to the sender, and each address verification is to be reported. A relay program might have ‘‘mlntviVGR.BRL.MIL*,’’ with ‘‘mlv’’ as above and the other settings indicating that mail for non-local channels is to be sent immediately, the author information is to be trusted, and the "Received: " component should cite the mail as being relayed via Internet host VGR.BRL.MIL.
c. To:	One address, terminated by a newline ('\n').
d. From:	Status character, from *mmdf.h,* plus human-oriented text plus newline.
e. Loop:	Back to (c). Terminate with address line having only an exclamation mark (!), with newline.
f. To:	Message text, in Internet RFC #822 format. Multi-line, terminated by null ('\0').

g. From:	Status character, text, newline.
h. Loop:	Back to (b). Terminate with initialization line having only a null, without newline.

Channels

When MMDF is used in conjunction with the DARPA domain nameserver system, a ''delay'' channel should be configured to allow queuing of addresses that fail verification temporarily due to nameserver failures (unavailability). Two other special channels that can be configured are the ''badusers'' and ''badhosts'' channels. Mail to unknown users or unknown hosts will be queued to these channels if they are configured. The bad channels have no special code associated with them. The channel configuration should reference whatever table and program is necessary to reach a smarter host which can deliver or forward the mail. The channel should have the ''host='' parameter set to this host name. The channel names given above are reserved.

Files

Numerous. Generally under the MMDF login directory.

See Also

send(ADM), mmdf(S), deliver(ADM)

sulogin

access single-user mode

Syntax

sulogin

Description

sulogin is automatically invoked by *init* when the system is first started. It prompts the user to type the root password to enter system maintenance mode (single-user mode) or to type CTRL-D for normal startup (multi-user mode). *sulogin* should never be directly invoked by the user.

Files

/bin/sulogin

See Also

init(M)

Value Added

sulogin is an extension of AT&T System V provided in Altos UNIX System V.

swap

swap administrative interface

Syntax

/etc/swap -a swapdev swaplow swaplen
/etc/swap -d swapdev swaplow
/etc/swap -l

Description

The *swap* command 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, e.g., **/dev/dsk/as0**. *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 routine **/etc/rc** when going into multiuser mode.

-d Delete the specified swap area. *swapdev* is the name of a block special device, e.g., **/dev/dsk/as0**. *swaplow* is the offset in 512-byte blocks into the device where the swap area should begin. 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.

Notes

No check is done to see if a swap area being added overlaps with an existing swap area or file system.

sync

updates the super-block

Syntax

sync

Description

sync executes the *sync* system primitive. If the system is to be stopped, *sync* must be called to ensure file system integrity. Note that *shutdown*(ADM) automatically calls *sync* before shutting down the system.

See Also

sync(S)

Standards Conformance

sync is conformant with:

AT&T SVID Issue 2, Select Code 307-127;
and The X/Open Portability Guide II of January 1987.

sysadmsh

menu driven system administration utility

Syntax

sysadmsh

Description

sysadmsh is an easy-to-use menu interface designed to provide novice users with the tools needed for day-to-day system administration of the UNIX system.

WARNING: *sysadmsh* does not replace the documentation. It provides an overview of available system administration features and a reminder of tasks which need to be performed regularly. An understanding of the *Installation Guide,* the *System Administrator's Guide,* and the *User's Guide* is necessary to use *sysadmsh.*

Usage

sysadmsh menus can be invoked by logging in as the super-user (root) and entering:

sysadmsh

at the shell prompt.

Once you are in *sysadmsh,* on-line instructions for its use may be obtained by selecting the <F1> key.

Some *sysadmsh* options must be run from the system console device. Some options must be run while in single user (system maintenance) mode. Check the documentation manual page referenced by the menu selection for more information.

Environment Variables

sysadmsh uses the following environment variables:

- **SCOLIB** is used to locate the tcap directory which contains various terminal-specific O/A files. The location procedure is:

the directory **.tcap/** *terminal type* is searched for in the user's home directory, if this does not exist then,

the directory **$(SCOLIB)** /tcap/ *terminal type* is searched for, if this does not exist then,

the directory .tcap/generic is searched for in the user's home directory, if this does not exist then,

The directory **$(SCOLIB)** /generic is searched for.

If the **SCOLIB** variable is not explicitly set then it defaults to /usr/lib/sco

- **SYSADM** is used to find the O/A prompt file **libstrs**, plus the menu, form and help files.

There are three environment variables which *sysadmsh* considers to locate the editor it calls. **SA_EDITOR** is tried first, if this is null then **VISUAL** is tried, then **EDITOR.**

If none of the editor environment variables are set then, one of the following editors is chosen: **/usr/bin/lyrix**, **/bin/vi** or **/bin/ed** (listed in decreasing preference).

The following additional environment variables are used:

SA_MAIL If not set, the default mailer is SCO Portfolio **email** if installed, or regular UNIX **mail(C)** if not.

SA_PRINT If not set, the default printer device is **/dev/lp**.

See Also

System Administrator's Guide
User's Guide
Installation Guide

acctcom(ADM), accton(ADM), asktime(ADM), at(C), badtrk(ADM), checklist(F), chgrp(C), chmod(S), chown(C), configure(ADM) copy(C), cron(C), csh(C), custom(ADM), df(C), diff(C), dircmp(C), disable(C), diskcmp(C), diskcp(C), dmesg(ADM), dos(C), dtype(C), du(C), enable(C), fdisk(ADM), find(C), finger(C), fixperm(ADM), format(C), fsck(ADM), grpcheck(C), init(M), kill(C), login(M), lp(C), lpadmin(ADM), lpstat(C), mail(C), mkdev(ADM), more(C), mount(ADM), netutil(ADM), ps(C), quot(C), shutdown(ADM), systemid(F), tar(C), umount(ADM), uuinstall(ADM), vi(C), wall(ADM), who(C), write(C)

Notes

A knowledge of *vi*(C) is assumed for file edit selections, although the SCO Lyrix® editor is used when available.

Acknowledgements

This utility takes its design from the SCO Lyrix Word Processing System.

Value Added

sysadmsh is an extension of AT&T System V provided in Altos UNIX System V.

sysdef

output values of tunable parameters

Syntax

/etc/sysdef [system_namelist [conf]]

Description

The *sysdef* command outputs the values of all tunable parameters. It generates the output by analyzing the named operating system file (*system_namelist*) and extracting the configuration information from the name list itself.

Files

/unix	default operating system file (where the system namelist is)
/etc/conf/*	default directory containing master files

See Also

nlist(S)

Diagnostics

internal name list overflow
If the master table contains more than an internally specified number of entries for use by *nlist*(S).

Standards Conformance

sysdef is conformant with:

AT&T SVID Issue 2, Select Code 307-127.

tcbck, smmck, authckrc

trusted computing base checker

Syntax

tcbck

Description

tcbck checks the files in the trusted computing base for files that were caught in the process of being updated when the system went down, and for files that have been removed. *tcbck* is invoked by the scripts */etc/smmck* during system maintenance mode, and by */etc/authckrc* when the system enters multi-user mode. The check proceeds as follows:

1. */etc/smmck* runs *tcbck* to clean up any database files that were left in an interim state while being updated (files are created with **-o** (old) and **-t** (new) suffixes, respectively). When this process is interrupted, -o and -t files are left and must be reconciled before the system will function properly. *tcbck* checks the **/etc/auth/system**, **/etc/auth/subsystems** and **/tcb/files/auth/*** directories. If there are multiple versions of a file, the extra files are removed. When a -t file is found, the following is displayed:

   `/etc/tcbck: file`*file* `missing, saved`*file*`-t as`*file*

   This message is repeated for all files found in that state in the specified directories.

2. Next *tcbck* removes **/etc/auth/system/pw_id_map** and **/etc/auth/system/gr_id_map** because the modification times of these files are compared with that of **/etc/passwd** and **/etc/group** and problems can occur when th system clock is reset.

3. **tcbck** checks that key system files are present and that they are not of zero length. If a file is missing (or zero length) then a message similar to this is displayed:

   `/etc/tcbck: file` *file* `is missing or zero length`

This process is repeated for each of the following files:

/etc/auth/system/default†
/etc/auth/system/files
/etc/auth/system/devassign
/etc/auth/system/ttys
/etc/auth/system/authorize†
/tcb/files/auth/r/root†
/etc/group
/etc/passwd†

When this process is complete, if any files were missing or -t files were substituted for real files, the following message is displayed:

```
/etc/smmck: restore missing files from backup or distribution.
```

4. If critical database files have been removed or corrupted (files marked with a dagger (†) in the previous file list are considered critical) then the system enters maintenance mode automatically without asking for the root password. If no critical database files were lost, the system prompts for maintenance mode or normal operation.

5. After the system goes to init level 2, */etc/authckrc* reinvokes *tcbck* to confirm that the files reported missing previously have been restored: Any missing files are listed, followed by this message:

```
/etc/authckrc: Log in on the OVERRIDE tty and restore
      the missing files from a backup or the distribution disks.
```

Missing files will have to be replaced when the system comes up multi-user.

6. Finally **authckrc** prompts for checking the protected subsystem databases. If the response is yes, the *authck*(ADM) program is run.

Value Added

tcbck is an extension of AT&T System V provided in Altos UNIX System V.

timex

time a command; report process data and system activity

Syntax

timex [options] command

Description

The given *command* is executed; the elapsed time, user time and system time spent in execution are reported in seconds. Optionally, process accounting data for the *command* and all its children can be listed or summarized, and total system activity during the execution interval can be reported.

The output of *timex* is written on standard error.

Options are:

-p List process accounting records for *command* and all its children. This option works only if the process accounting software is installed. Suboptions **f**, **h**, **k**, **m**, **r**, and **t** modify the data items reported. The options are as follows:

-f Print the *fork*/*exec* flag and system exit status columns in the output.

-h Instead of mean memory size, show the fraction of total available CPU time consumed by the process during its execution. This "hog factor" is computed as:

(total CPU time)/(elapsed time).

-k Instead of memory size, show total kcore-minutes.

-m Show mean core size (the default).

-r Show CPU factor (user time/(system-time + user-time).

-t Show separate system and user CPU times. The number of blocks read or written and the number of characters transferred are always reported.

-o Report the total number of blocks read or written and total characters transferred by *command* and all its children. This option works only if the process accounting software is installed.

-s Report total system activity (not just that due to *command*) that occurred during the execution interval of *command*. All the data items listed in *sar*(C) are reported.

See Also

sar(ADM).

Warning

Process records associated with *command* are selected from the accounting file **/usr/adm/pacct** by inference, since process genealogy is not available. Background processes having the same user-id, terminal-id, and execution time window will be spuriously included.

Examples

A simple example:

```
timex -ops sleep 60
```

A terminal session of arbitrary complexity can be measured by timing a sub-shell:

```
timex -opskmt sh
```

session commands
EOT

Standards Conformance

timex is conformant with:

AT&T SVID Issue 2, Select Code 307-127.

tplot

graphics filters

Syntax

tplot [**-T**terminal [**-e** raster]]

Description

This command reads plotting instructions [see *plot*(F)] from the standard input and in general produces, on the standard output, plotting instructions suitable for a particular *terminal*. If no *terminal* is specified, the environment parameter **$TERM** [see *environ*(M)] is used. Known *terminal*s are:

300
DASI 300.

300S
DASI 300s.

450
DASI 450.

4014
Tektronix 4014.

ver
VERSATEC D1200A. This version of *plot* places a scan-converted image in **/usr/tmp/raster$$** and sends the result directly to the plotter device, rather than to the standard output. The **-e** option causes a previously scan-converted file *raster* to be sent to the plotter.

Files

/usr/lib/t300
/usr/lib/t300s
/usr/lib/t450
/usr/lib/t4014
/usr/lib/vplot
/usr/tmp/raster$$

See Also

plot(S), plot(F), term(M)

uadmin

administrative control

Syntax

/etc/uadmin command function

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 *command* and *function* are converted to integers and passed to the *uadmin* system call.

See Also

uadmin(S)

uconfig

UNIX configuration manager

Syntax

/altos/bin/uconfig [-dltpDiyM] [**-r** *root*] [**-m** *mdevice*] [**-s** *sdevice.d*] [**-e** *meisa*]

Description

uconfig is the top-level kernel configuration utility for ISA (PC/AT) and EISA bus systems. It is used to generate a new UNIX kernel, and perform other related system configuration tasks. On an EISA system, *uconfig* automatically configures the kernel to match the hardware.

For an EISA system, *uconfig* must be run each time the system hardware is changed (for example, whenever an expansion board is added or removed, or an expansion board is moved from one EISA slot to another), *uconfig* should be invoked after first running the EISA Configuration Utility. (The EISA Configuration Utility is provided with your computer system on a separate Configuration diskette. Its use is described in the Owner's Guide for your system.)

uconfig performs the following tasks:

1. Modifies the kernel and device driver configuration files to match the hardware (EISA systems only).

2. If an EISA hardware function exists but no mapping to an associated device driver exists in the **meisa**(F) file, *uconfig* prompts you for the driver name, and updates the **meisa** file accordingly. (**Note**: If no driver name is entered at this prompt, but only the Return key is pressed, *uconfig* ignores the unmapped function, and updates the **meisa** file so that this function is automatically ignored the next time *uconfig* is invoked.)

3. Runs inittab and node setup shell scripts (see **meisa**(F) and script specifications below) (EISA systems only).

4. Sets up the **slot_info** file for use by *pcu*(ADM), the Port Configuration Utility (EISA systems only).

5. Optionally (with **-M**), adjusts the kernel's tunable parameters to match the system memory size.

6. Re-links the kernel.

7. Sets up the terminal database used by the system security services.

After *uconfig* re-links the kernel, you must re-boot the system to activate the new kernel.

If the **ROOT** environment variable is set to a valid path name, then all absolute path names are prefixed with the **ROOT** path. If the **ROOT** environment variable is undefined, and the **-r** option is not used, then the default is / (the root directory).

Options

-d Display hardware information only; no kernel configuration is done.

-l Suppress linking the kernel. However, all kernel files are still configured.

-t Suppress setting up the protected terminal database.

-p Suppress setting up the *pcu* **slot_info** file (EISA systems only).

-D Debugging mode. Produce verbose diagnostic messages.

-i Ignore all EISA hardware functions that are not mapped to a device driver, that is, suppress asking for a driver name for such functions. Also, these functions are not added to the **meisa** file (EISA systems only).

-y If the *root* directory is '/', assume 'y' to the questions that normally appear (that is, "Should the kernel environment be built?" and "Use this as the default kernel on all boots?"). These questions are not displayed.

-M Invoke *idmemtune*(ADM) to adjust the kernel tunable parameters to match the system memory size.

-r *root*
Specify an alternate *root* directory, below which other absolute path names are derived. This option overrides the **ROOT** environment variable.

-m *mdevice*
Specify an alternate *mdevice* file (the default file is *root*/**etc/conf/cf.d/mdevice**).

-s *sdevice.d*
Specify an alternate *sdevice.d* directory (the default directory is *root*/**etc/conf/sdevice.d**).

-e *meisa*
Specify alternate *meisa* file (default is *root*/**etc/conf/cf.d/meisa**).

Set-up Shell Scripts

On an EISA system, *uconfig* performs extra system configuration tasks by invoking special shell scripts as described in this section.

If specified in the **meisa**(F) file, *uconfig* will invoke the following Bourne Shell [*sh*(C)] scripts:

root/**etc/conf/pack.d/***xxx*/*xxx***init**
root/**etc/conf/pack.d/***xxx*/*xxx***node**

where *xxx* is the driver name specified in the **meisa**(F) file. The *xxx***init** scripts are used to update the **/etc/conf/init.d/***xxx* files (which are used to create the **/etc/inittab** file, as described in *idmkinit*(ADM)). The *xxx***node** scripts are used to update the **/etc/conf/node.d/***xxx* files (which are used to create the special **/dev** files, as described in *idmknod*(ADM)). These scripts will be run as many times as the associated entries in the **sdevice**(F) file. The supplied arguments are: *lineno id type*.

The *lineno* argument is the relative position of the driver's entry in the **sdevice.d/***xxx* file, starting at 0. The *id* argument is the EISA board ID, and *type* is the EISA function type string associated with the driver in the particular instance. These arguments may be used to determine, for example, which **tty** entries to make for a multi-port serial port expansion board.

The *xxx***node** scripts, in addition, can echo two numbers (for example, **echo "$num1 $num2"**) which are passed back into *uconfig* and recorded in the **slot_info** file for use by *pcu*(ADM). The two numbers *$num1* and *$num2* should define a range to be used for tty numbers on a serial port expansion board.

These shell scripts must not echo anything to *stdout* except as noted above. All error diagnostics should be directed to *stderr*. In addition, the exit status of the scripts must be 0 (zero) to indicate success, and non-zero to indicate failure.

During the time when these shell scripts are invoked, the old *root*/**etc/conf/init.d/***xx* and *root*/**etc/conf/node.d/***xx* files, if they existed, are saved as *root*/**etc/conf/init.d/***xx***.sav** and *root*/**etc/conf/node.d/***xx***.sav**. These older files can be used by the shell scripts, if necessary, as a basis for building the new file. After all the scripts are run, the saved files are removed.

Files

*/etc/conf/cf.d/meisa
/etc/conf/sdevice.d/
*/etc/slot_info

See Also

meisa(F), sdevice(F), idmemtune(ADM)

Value Added

uconfig is an extension to AT&T UNIX System V provided in Altos UNIX System V.

umount

dismounts a file structure

Syntax

/etc/umount special-device

Description

umount announces to the system that the removable file structure previously mounted on device *special-device* is to be removed. Any pending I/O for the file system is completed, and the file structure is flagged clean. For a detailed explanation of the mounting process, see *mount*(ADM).

Files

/etc/mnttab Mount table

See Also

mount(ADM), mount(S), mnttab(F)

Diagnostics

device busy	An executing process is using a file on the named file system.

Standards Conformance

umount is conformant with:

AT&T SVID Issue 2, Select Code 307-127;
and The X/Open Portability Guide II of January 1987.

upsconfig

UPS shutdown configuration utility

Syntax

upsconfig [**-i**] [**-p**] [**-t***pwrtype*] [**-f***failtime*] [**-c***pwrcnt*] [**-u***upstime*] [**-w***pwrtime*] [**-e***termtime*] [**-d***maj,min*]

Description

The *upsconfig* utility lets you change the configurable parameter settings that control how your system reacts in an uninterruptible power supply (UPS) power failure condition. Several parameter settings can be changed. The options to *upsconfig* are:

-i Set parameters according to values found in the file /etc/**upsparam.**

-p Display the current settings of all the configurable parameters.

-t*pwrtype* Set the UPS shutdown mode. *pwrtype* can be **shutkill** or **shutsave.** See the description of these two modes in the "Shutdown Modes" section below.

-f*failtime* The time in seconds to wait before checking again for a second power failure condition after the first power failure is detected. This setting is used to distinguish momentary power interruptions from total power failures. The default setting is 10.

-c*pwrcnt* The maximum number of power failure interrupts that can occur within the time interval *failtime*, after which the power source is considered unreliable. The default setting is 30.

-u*upstime* The time in seconds that the UPS battery backup unit can operate reliably after power has been turned off. The default setting is 180.

-w*pwrtime* The time in seconds for the system to wait after posting the SIGPWR signal to all processes before initiating shutdown procedures. The default setting is 30.

-e*termtime* The time in seconds for the system to wait after posting the SIGTERM signal to all processes before posting the SIGKILL signal to all processes. The default setting is

20. This option is meaningful only if the shutkill mode is selected.

-d*maj,min* If the shutsave mode is selected, this option specifies the restart disk device in which the system memory image is to saved before the UPS turns its battery off. The major and minor device numbers, spearated by commas, are used to identify this hard disk area. Please note that a restart disk division must be allocated using *divvy*(ADM) before you can invoke *upsconfig* with this option. The disk device must be named **restart** and **rrestart** for the block and character special device files, respectively. The default major and minor device numbers are derived from the **/dev/restart** device, and should be ususally accepted.

If *upsconfig* is invoked without any options, you are prompted for each of the settings described above. A null response followed by ⟨Return⟩ will leave the current configuration value the same.

The **-p** option displays the current settings of all the configurable parameters described above. No other options are allowed to be given with the **-p** or **-i** options.

A sanity check is performed on all of the values you enter. If the shutdown mode is shutkill, the total times of *failtime*, *pwrtime*, and the estimated disk output time cannot exceed the value of *upstime*. The estiamted disk output time will be displayed if the selected shutdown mode is shutsave, and the **-p** option is given. If there are any inconsistencies, the following error message is displayed:

```
Could not change parameters: Invalid argument
```

Only the super-user is allowed to change any of these configurable parameters.

Shutdown Modes

There are two shutdown modes that can be selected: **shutkill** or **shutsave.** If the system is connected to a supported UPS, the following shutdown sequences take place after a power failure is detected, depending on which mode you have selected::

shutkill SIGPWR is sent to all processes, and then followed by SIGTERM and SIGKILL. The buffer cache is then flushed and all filesystems unmounted. The system then shuts off the UPS battery. When power is restored, the system performs a normal bootstrap to multiuser mode.

shutsave SIGPWR is sent to all processes, and then the system memory image, values of registers, and other machine states are all saved to the hard disk "restart device." The system then shuts off the UPS battery. When power returns, the computer is restored to the saved state, allowing the system to restart and resume all processes as if power never halted.

Notes

The UPS parameters configured with *upsconfig* are saved across system reboots in the **/etc/upsparam** file. When *upsconfig* changes the parameters in the kernel, it also writes these values to the **/etc/upsparam** file. When the system is rebooted, the **/etc/bcheckrc** file executes *upsconfig* with the **-i** option, re-initializing the parameters in the kernel to the configured values.

See Also

upscfg(S), "Uninterruptible Power Supply Support" in the *System Administrator's Guide*.

Value Added

upsconfig is an extension to AT&T System V provided in Altos UNIX System V.

uucheck

checks the uucp directories and permissions file

Syntax

/usr/lib/uucp/uucheck [**-v**] [**-x** debug_level]

Description

uucheck checks for the presence of the *uucp* system required files and directories. 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 *uucheck* 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/Maxuuxqts
/usr/spool/uucp/*
/usr/spool/uucppublic/*

See Also

uucico(ADM), uusched(ADM), uucp(C), uustat(C), uux(C)

Notes

The program does not check file/directory modes or some errors in the Permissions file such as duplicate login or machine name.

uucico

file transport program for the UUCP system

Syntax

/usr/lib/uucp/uucico [**-r** role_number] [**-x** debug_level]
[**-i** interface] [**-d** spool_directory] [**-s**] [**-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, *uusched* , 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 debugging.

The **-i** option defines the interface used with *uucico*. This interface only affects slave mode. Known interfaces are UNIX (default), TLI (basic Transport Layer Interface), and TLIS (Transport Layer Interface with Streams modules, read/write); only the default, UNIX, is applicable in this release.

The **-d** option can be used to specify the *spool* directory: the default is */usr/spool/uucp*.

If **-s** is specified, a call to the specified site is made even if there is no work for site *sitename* in the spool directory, but call only when times in the **Systems** file permit it. This is useful for polling sites that do not have the hardware to initiate a connection.

The **-S** option can be used to specify the system name, overriding the call schedule given in the **Systems** file. For example, **-S** can be used to call a system which is said to be "Never" called in the *Systems* file.

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/uucppublic/*

See Also

uusched(ADM), uutry(ADM), cron(C), uucp(C), uustat(C), uux(C)

uuclean

UUCP spool directory clean-up

Syntax

/usr/lib/uucp/uuclean [-Ctime] [**-D**time] [**-W**time] [**-X**time]
[**-m**string] [**-o**time] [**-s**system] [-**x**debug_level]

Description

uuclean 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 waiting for a given number of days (default 1). Note that *uuclean* will process as if all option *times* were specified to the default values unless *time* is specifically set.

The following options are available.

-C*time* Any **C.** files greater or equal to *time* days old will be removed with appropriate information to the requestor. (default 7 days)

-D*time* 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)

-W*time* 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 *JOBID*, 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)

-X*time* 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)

-m*string* This line will be included in the warning message generated by the **-W** option.

-o*time* Other files whose age is more than *time* days will be deleted. (default 2 days) The default line is "See your local administrator to locate the problem".

-s*system* Execute for *system* spool directory only.

-x*debug_level*
The **-x** debug level is a single digit between 0 and 9; higher numbers give more detailed debugging information.

This program is typically started by the shell *uudemon.clean*, which should be started by *cron*(C). *uuclean* can only be executed by the super user or uucp.

Files

/usr/lib/uucp	directory with commands used by *uuclean* internally
/usr/spool/uucp	spool directory

See Also

cron(C), uucp(C), uux(C)

uudemon: uudemon.admin, uudemon.clean, uudemon.hour, uudemon.poll, uudemon.poll2

UUCP administrative scripts

Description

UUCP communications and file maintenance can be automated with the use of the **uudemon.hour**, **uudemon.poll**, **uudemon.poll2**, **uudemon.admin**, and **uudemon.clean** shell scripts. While in multi-user mode, **cron** scans files in **/usr/spool/cron/crontabs** once each minute for entries to execute at this time. An example crontabs file, *crontab.eg*, is provided to activate these daemons. The system administrator should copy these from **/usr/lib/uucp** to **/usr/spool/cron/crontabs/uucp**. To do this, log in as user **uucp**, edit the **crontab.eg** file to make any changes, and then enter the following command:

crontab crontab.eg

This will replace the original crontab entry.

uudemon.admin

The **uudemon.admin** shell script, as delivered, runs the **uustat** command with **-p** and **-q** options. The **-q** reports on the status of work files (C.), data files (D.), and execute files (X.) that are queued. The **-p** prints process information for networking processes listed in the lock files (/usr/spool/locks). It sends resulting status information to the UUCP administrative login (**uucp**) via mail.

The default crontab entry for **uudemon.admin** is:

```
48 10,14 * * 1 - 5 /bin/su uucp -c \
     "/usr/lib/uucp/uudemon.admin" > /dev/null
```

uudemon.clean

The **uudemon.clean** shell script, as delivered, takes log files for individual machines from the /usr/spool/.Log directory, merges them, and places them in the /usr/spool/.Old directory with other old log information. If log files get large, the ulimit may need to be increased. It also removes work files (C.) 7 days or older, data files (D.) 7 days old or older, and execute files (X.) 2 days old or older from the spool files. **uudemon.clean** mails a summary of the status information gathered during the current day to the UUCP administrative login (uucp).

The default crontab entry for **uudemon.clean** is:

```
45 23 * * * ulimit 5000; /bin/su uucp -c \
        "/usr/lib/uucp/uudemon.clean" > /dev/null
```

uudemon.hour

The **uudemon.hour** shell script calls the **uusched** program to search the spool directories for work files (C.) that have not been processed and schedules these files for transfer to a remote machine. It then calls the **uuxqt** daemon to search the spool directories for execute files (X.) that have been transferred to your computer and were not processed at the time they were transferred.

This is the default root *crontab* entry for **uudemon.hour** :

```
39, 9 * * * * /usr/lib/uucp/uudemon.hour > /dev/null
```

This script runs twice per hour (at 39 and 9 minutes past).

uudemon.poll

uudemon.poll uses the **Poll** (or the alternative **Poll.hour** and **Poll.day**) file (see *poll*(F)) for polling remote computers. The **uudemon.poll** script controls polling but does not actually perform the poll. It merely sets up a polling file (**C.sysnxxxx**) in the /usr/spool/uucp/*nodename* directory, where *nodename* is replaced by the name of the machine. This file will in turn be acted upon by the scheduler (started by **uudemon.hour**). The **uudemon.poll** script is scheduled to run twice an hour just before **uudemon.hour** so that the work files will be there when **uudemon.hour** is called. The default root crontab entry for **uudemon.poll** is as follows:

```
1,30 * * * * "/usr/lib/uucp/uudemon.poll > /dev/null"
```

uudemon.poll2 is an alternative to **uudemon.poll**, which uses a different scheme and different poll files. Listing a site in the **Poll** file gives you control over the lower bound on number-of-calls-per-day (at least as many as you specify in **Poll**), but still no control on the upper bound. (This is because **uudemon.poll** uses the the time field of the **Systems** file, which is not suited to the purposes of polling). **uudemon.poll2** permits much more precise control of scheduling. To use **uudemon.poll2**, you must remove the call to *uusched* from **uudemon.hour**, and run **uudemon.poll2** in place of **uudemon.poll** from *cron*. **uudemon.poll2** reads **Poll.hour** (or **Poll.day** if called with the **-d** option) to determine whom to poll much like **uudemon.poll**, but calls *uucico* directly, using the **-S** option, thus overriding the time field of the **Systems** file.

Files

/usr/lib/uucp/Systems
/usr/lib/uucp/uudemon.admin
/usr/lib/uucp/uudemon.clean
/usr/lib/uucp/uudemon.hour
/usr/lib/uucp/uudemon.poll
/usr/lib/uucp/uudemon.poll2
/usr/lib/uucp/Poll
/usr/lib/uucp/Poll.hour
/usr/lib/uucp/Poll.day

See Also

uusched(ADM) uucico(ADM), uuclean(ADM), cron(C), uucp(C), poll(F) systems(F)

Standards Conformance

uudemon is conformant with:

AT&T SVID Issue 2, Select Code 307-127.

uudemon.poll2 is an extension of AT&T System V provided in Altos UNIX System V.

uugetty

set terminal type, modes, speed, and line discipline

Syntax

/usr/lib/uucp/uugetty [**-t** *timeout*] [**-r**] *line* [*speed* [*type* [*linedisc*]]]
/usr/lib/uucp/uugetty -c *file*

Description

uugetty is a standard *getty*(M) modified to allow a tty line to be used by *uucico*, *cu*, and *ct*; that is, the line can be used in both directions. The *uugetty* 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. Note that in the **-r** case, 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:

```
"" \r\d\r\d\r\d\r in:--in: ...
```

where the ... is whatever would normally be used for the login sequence.

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 or direct line:

```
30:2:respawn:/usr/lib/uucp/uugetty -r -t 60 tty00 1200
```

The meanings of the available options are:

-t *timeout*
Specifies that *uugetty* should exit if the open on the line succeeds and there is no response to the login prompt in *timeout* seconds. *timeout* is replaced by an integer.

-r Causes *uugetty* to wait to read a character before it puts out the login message, thus preventing two *uugetty*s from looping. An entry for an intelligent modem or direct line that has a *uugetty* on each end must use this option.

line

Defines the name of the line to which *uugetty* will attach itself. The line name will point to an entry in the **/dev** directory. For example, **/dev/tty00**.

speed

Defines the entry to use from the **/etc/gettydefs** file. The entry defines the line speed, the login message, the initial tty setting, and the next speed to try if the user says the speed is inappropriate (by sending a *break* character). The default *speed* is 300.

type

Defines the type of terminal connected to the line. The default terminal is **none**, representing a normal terminal unknown to the system.

linedisc

Sets the line discipline to use on the line. The default is **LDISC0**, which is the only one currently compiled into the operating system.

-c *file*

Checks the speed and tty definitions in *file* and sends the results to standard output. Unrecognized modes and improperly constructed entries are reported. For correct entries, flag values are printed. *file* is replaced by **/etc/gettydefs** or a similarly structured file.

Files

/etc/gettydefs
/etc/issue

See Also

login(C), ct(C), cu(C), getty(M), init(M), uucico(ADM), tty(HW), ioctl(S), gettydefs(F), inittab(F)

Notes

ct will not work when *uugetty* is used with an intelligent modem such as penril or ventel.

uuinstall

administers UUCP control files

Syntax

/etc/uuinstall [-r]

Description

The *uuinstall* program is used to manage the content of the control files used by the *uucp* communications system. It allows the user to change the contents of these files without using a text editor. The user need not know the detailed format of each of the control files, although he must be familiar with the function of the various fields within the files. These details are explained in the *System Administrator's Guide* .

The *uuinstall* program can only be executed by the super-user. When invoked with the optional **-r** flag, *uuinstall* will not allow any of the files to be modified whether or not the user has made changes to the files.

If *uuinstall* finds any of the required **uucp** control files missing from the system, it will create them with the correct access permissions and ownership.

Files

/etc/systemid
/usr/lib/uucp/Systems
/usr/lib/uucp/Permissions
/usr/lib/uucp/Devices

See Also

mkuser(ADM)

uulist

converts a UUCP routing file to MMDF format

Syntax

/usr/mmdf/table/tools/uulist

Description

uulist is a conversion utility to produce MMDF-compatible UUCP routing files from the UUCP routing file.

After installing MMDF with *custom*, restore **/usr/lib/uucp/Systems** from backup media. Log in as *root* and run the conversion script */usr/mmdf/table/tools/uulist* from the **/usr/mmdf/table** directory. You now have UUCP domain and channel files, **uucp.dom** and **uucp.chn**, in the current directory. Use the *chown* command to make these files owned by *mmdf*. Log out from the super user account.

After creating these files in **/usr/mmdf/table**, you must rebuild the MMDF hashed database. Log in as *mmdf* and run *dbmbuild* from **/usr/mmdf/table**.

Files

/usr/lib/uucp/Systems
/usr/mmdf/table/uucp.chn
/usr/mmdf/table/uucp.dom

See Also

dbmbuild(ADM), tables(F), "Setting Up Electronic Mail" in the *System Administrator's Guide*

Value Added

uulist is an extension of AT&T System V provided in Altos UNIX System V.

uusched

the scheduler for the UUCP file transport program

Syntax

/usr/lib/uucp/uusched [**-x** debug_level] [**-u** debug_level]

Description

uusched is the *uucp* file transport scheduler. It is usually started by the daemon *uudemon.hour* that is started by *cron*(C) from an entry in */usr/spool/cron/crontabs/root*:

```
39,9 * * * * /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 *uusched* 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/lib/uucp/Maxuuscheds
/usr/spool/uucp/*
/usr/spool/uucppublic/*

See Also

uucico(ADM), cron(C), uucp(C), uustat(C), uux(C)

uutry

tries to contact remote system with debugging on

Syntax

/usr/lib/uucp/uutry [**-x** debug_level] [**-r**] system_name

Description

The *uutry* program is a shell that invokes *uucico* to call a remote site. Debugging is automatically enabled at default level 5; **-x** overrides this value. If *uutry* successfully connects to the remote system, *uutry* stores the debugging output in the file **/tmp/***system*, where *system* is the name of the remote system. In addition, *uutry* uses *tail* **-f** to print the last 10 lines of the debugging output to the standard output.

To break out of the shell created by *uutry* , press DELETE or BREAK . This returns control to the terminal while *uucico* continues to run, sending the output to **/tmp/***system_name*.

The **-r** option overrides the retry time in **/usr/spool/uucp/.status.**

Files

/usr/lib/uucp/Systems
/usr/lib/uucp/Permissions
/usr/lib/uucp/Devices
/usr/lib/uucp/Maxuuscheds
/usr/lib/uucp/Maxuuxqts
/usr/spool/uucp/*
/usr/spool/uucppublic/*
/tmp/system_name

See Also

uucico(ADM), uucp(C), uux(C)

uuxqt

executes remote command requests

Syntax

/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 execution 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/*

See Also

uucico(ADM), uucp(C), uustat(C), uux(C), mail(C)

vddaemon

virtual disk initialization

Syntax

vddaemon

Description

The virtual disk daemon, *vddaemon*, is started automatically by *init* whenever the operating system starts. This daemon initializes the system using information found in **/etc/vdtab**. It then sleeps in the background, keeping the physical devices open for the virtual I/O.

Files

/dev/vd*,
/dev/rvd*,
/etc/vdtab,
/etc/vdbadtab

See Also

add.vd(ADM), del.vd(ADM), vdinfo(ADM), vdutil(ADM)

Value Added

vddaemon is an extension of AT&T System V provided in Altos UNIX System V.

vdinfo

display virtual disk information

Syntax

vdinfo [**-v**] [**-k** | **-b** *dev_name*] | [**-n** *vd_num*]

Description

The *vdinfo* utility displays the current virtual disk set-up.

Options

The default option is to print information on all virtual disks currently set up in the system. See "Usage" below for details on output produced by the default (no optional arguments) of *vdinfo*.

-v Enable verbose mode.

-k Display the size of disk *dev_name* in 512-byte blocks. The specified disk may be a physical or virtual disk, and must be identified by its "raw" (character special) device filename (e.g., **/dev/rhdb0** for the second physical hard (disk 1), or **/dev/rvd0** for virtual disk 0).

-b Display the following information about the specified device name, *dev_name*, in this order: character or block file, major number, minor number, physical disk number, "virtual" drive number and division number. Note that the "virtual" drive number in this sense means the hard disk partition number.

-n Display information for virtual disk number *vd_num* only.

Usage

The default option with the verbose option only (i.e., **vdinfo -v**) displays information on each virtual disk in columns. Displaying all the existing virtual disks might take a few seconds since it has to search through **/dev** to gather all the physical device names. The time it takes depends on the number of virtual disks in the system.

The default option without the verbose option (i.e., simply **vdinfo**) displays information about all the existing virtual disks. Each line represents a single striped or mirrored virtual disk.

Here is the format of the default output produced by *vdinfo*:

vdnum **S** *stripsz phys_dev* ... [: *mvdnum* **M** *mstripsz mphys_dev* ...]

where:

vdnum
represents the primary virtual disk number.

S indicates striping, which is always present if the system contains a virtual disk. Virtual disks that are mirrored only also contain this S character, since they are considered "striped" between the primary virtual disk and the mirror disk. See "Usage" below.

stripsz
represents the stripe size (in 1024-byte blocks) of the primary virtual disk.

phys_dev
is a list of the physical devices that comprise the primary virtual disk.

: is an optional character that, if present, indicates that this virtual disk is also striped. The information following the colon describes the mirror part(s) of the virtual disk. Otherwise, this virtual disk is striped only.

mvdnum
represents the mirror virtual disk number.

M represents mirroring.

mstripsz
represents the stripe size (in 1024-byte blocks) of the mirror disk.

mphys_dev
is a list of the physical devices that comprise the mirror disk.

Note that the only difference between a "mirrored" disk and a "striped and mirrored" disk is that a mirrored disk has only one device in the primary virtual disk. Otherwise they are exactly the same.

Examples

The following default *vdinfo* output (with no options) indicates that the there is only one virtual disk (**/dev/vd0**, or simply virtual disk 0), and that it is striped between physical devices **/dev/hdb10**, **/dev/hdc10**, and **/dev/hdd10**, with a stripe size of 16 KB:

```
% vdinfo
0 S 16 /dev/hdb10 /dev/hdc10 /dev/hdd10
```

Using **vdinfo -b** as shown below produces output that indicates the device name **/dev/hdc0** is a block special device file with a major number of 64 and a minor number of 64, and that it is located at physical disk number 2, partition ("virtual disk") number 0 (all partitions on the disk), and division number 0 (the first division).

```
% vdinfo -b /dev/hdc0
b 64 64 1 0 0
```

The following shows how you can use **vdinfo -b** with a virtual disk device name instead of a physical device name as used in the above example. Note also that the "raw" device name is used, thus displaying the character "**c**" file data.

```
% vdinfo -b /dev/rvd0
c 25 0 0 0 0
```

The following *vdinfo* output shows how virtual disk 1 (**/dev/vd0**) is set up. The two *vdinfo* commands are shown to illustrate the effect of the **-v** "verbose" option (remember that the "virtual" label indicates the partition number):

```
% vdinfo -b /dev/vd0
b 25 0 0 0 0

% vdinfo -v -b /dev/vd0
/dev/vd0 (25,0): block device, physical = 0, virtual = 0, divvy = 0
```

The following *vdinfo* command illustrates the use of the **-k** option with the virtual disk number 0. Note that the **-k** option requires the character-based special device filename (**rvd0**):

```
% vdinfo -k /dev/rvd0
516000
```

You can also use **vdinfo -b** to learn the sizes of the individual filesystems that make up the virtual disk:

```
% vdinfo -k /dev/rhdb10
174078

% vdinfo -k /dev/rhdc10
172030

% vdinfo -k /dev/rhdd10
174002
```

The following *vdinfo* output indicates that the first virtual disk (virtual disk 0) is both striped and mirrored. The primary virtual disk is striped between physical devices **/dev/hdb10** and **/dev/hdc10**, with a stripe size of 16 KB. The primary disk is mirrored with the second virtual disk (**/dev/vd1**, or simply virtual disk 1). This disk is striped between physical devices **/dev/hdd10** and **/dev/hde10**, with a stripe size of 16 K:

```
0 S 16 /dev/hdb10 /dev/hdc10 : 1 M 16 /dev/hdd10 /dev/hde10
```

The following *vdinfo* output indicates that virtual disk 0 (the first virtual disk) is mirrored only (with virtual disk 1). Note that there is only one physical device, **/dev/hdb10**, in the primary virtual disk; this indicates that the primary virtual disk is mirrored only, and not striped. The 'S' character and stripe sizes values (in this case 16 for both primary and mirror disks) are always present on virtual disks.

This output line specifically indicates that virtual disk 0 (physical device **/dev/hdb10**) is mirrored with virtual disk 1 (physical device **/dev/hdc10**).

```
0 S 16 /dev/hdb10 : 1 M 16 /dev/hdc10
```

Files

/dev/vd*,
/etc/rvd*,
/etc/vdtab

See Also

add.vd(ADM),
del.vd(ADM),
vddaemon(ADM)
vdutil(ADM)

Notes

Physical device names are obtained by matching the device numbers with files that appear in the */dev* directory. If the name is not found, then the major and minor device number pair is displayed instead.

Value Added

The *vdinfo* utility is an extension of AT&T System V provided in Altos UNIX System V.

vdutil

virtual disk utility

Syntax

/altos/bin/vdutil

Description

vdutil is a menu-driven program that performs maintenance tasks required to display, add, delete, or fix striped and mirrored virtual disks. *vdutil* provides a front-end interface for various other virtual disk commands and files, including *vdinfo*, *add.vd*, *del.vd*, and *vdbadblk*. You should use *vdutil* for all virtual disk administration tasks.

vdutil performs five basic tasks:

- display virtual disk information
- add a virtual disk
- remove a virtual disk
- fix a mirrored virtual disk
- display any bad physical blocks, or components of a mirrored virtual disk

When adding, deleting or fixing a virtual disk, the system must be in single-user mode. However, you may use *vdutil* to display virtual disk information while the system is in any mode.

Detailed descriptions of these five tasks are provided in the following sections.

Display Virtual Disk Information

Virtual disk information may be displayed for a specific virtual disk, or for all existing virtual disks. The displayed information includes the following:

Whether the disk is a mirrored or striped disk.

The underlying components (which may be physical disk divisions, or other virtual disks).

The block size (in 512 byte blocks).

Add a Virtual Disk

When adding a virtual disk, *vdutil* asks several questions. Refer to *System Administrator's Guide* for complete information on this procedure.

Delete a Virtual Disk

To delete a virtual disk, the user is asked to specify which virtual disk should be removed. *vdutil* will then remove the associated devices, delete the entry from **/etc/vdtab**, and (if necessary) delete any associated entry in the mount table.

Fix a Mirrored Virtual Disk

When selecting this option of *vdutil*, you will be asked which virtual disk to repair. You may specify any valid mirrored disk (or **a** for all mirrored disks). The remainder of this procedure is then automated. If any physical component has recorded just a few bad blocks, then those bad blocks will be repaired (using *badtrk(ADM)*), and the good data will be restored. If there have been "too many" errors, the whole physical component will have been disabled, and some external method of repair (or replacement) should be used. If this has already been done, then the whole component will be re-mirrored, and the component will be brought back into service. Note that in this case, the copying of the good data may take some time.

Display Bad Block Information on a Mirrored Virtual Disk

When selecting this option of *vdutil*, you will be asked which virtual disk to display. You may specify any valid mirrored disk (or **a** for all mirrored disks). Any corresponding bad block information will be displayed. If a physical component has been disabled due to "too many" errors, then this will be noted also. If so, then external action should be taken to remedy this situation, before selecting the option to Fix a Virtual Disk.

See Also

add.vd(ADM), del.vd(ADM), vdinfo(ADM), ''Virtual Disks'' in the *System Administrator's Guide*

Value Added

vdutil is an extension of AT&T System V provided in Altos UNIX System V.

vectorsinuse

displays the list of vectors currently specified in the sdevice file

Syntax

/etc/conf/cf.d/vectorsinuse

Description

This script searches the **sdevice** file and displays a list of the interrupt vectors already in use.

You must move to the **/etc/conf/cf.d** to execute *vectorsinuse*.

When installing a device driver with the Link Kit, you can use *vectorsinuse* to find an available interrupt vector for the driver. When you invoke the *configure* program to modify the system configuration files with the new driver information, use the **-v** option to indicate the vectors on which this device interrupts.

The -V option to *configure* performs a function similar to that of *vectorsinuse*. You specify a particular vector on which the device is capable of interrupting (refer to the device's hardware manual), and *configure* tells you if another device is already using that interrupt vector.

Files

/etc/conf/cf.d/sdevice

See Also

configure(ADM), sdevice(F), "Adding Device Drivers with the Link Kit" in the *System Administrator's Guide*

Value Added

vectorsinuse is an extension to AT&T System V provided in Altos UNIX System V.

volcopy

make literal copy of UNIX filesystem

Syntax

/etc/volcopy [options] fsname srcdevice volname1 destdevice volname2

Description

The *volcopy* command makes a literal copy of the UNIX filesystem using a blocksize matched to the device. *Options* are:

-a invoke a verification sequence requiring a positive operator response instead of the standard 10-second delay before the copy is made
-s (default) invoke the **DEL if wrong** verification sequence.

The program requests length and density information if it is not given on the command line or is not recorded on an input tape label. If the filesystem is too large to fit on one reel, *volcopy* will prompt for additional reels. Labels of all reels are checked. Tapes may be mounted alternately on two or more drives. If *volcopy* is interrupted, it will ask if the user wants to quit or wants a shell. In the latter case, the user can perform other operations (e.g., *labelit*) and return to *volcopy* by exiting the new shell.

The *fsname* argument represents the mounted name (e.g., **root, u1,** etc.) of the filsystem being copied.

The *srcdevice* or *destdevice* should be the physical disk section or tape (e.g.: **/dev/dsk/0s1** etc.).

The *volname* is the physical volume name (e.g.: **pk3, t0122,** etc.) and should match the external label sticker. Such label names are limited to six or fewer characters. *volname* may be - to use the existing volume name.

srcdevice and *volname1* are the device and volume from which the copy of the filesystem is being extracted. *destdevice* and *volname2* are the target device and volume.

fsname and *volname* are recorded in the last 12 characters of the super block (**char fsname[6], volname[6];**).

Files

/etc/log/filesave.log a record of filesystems/volumes copied

See Also

labelit(ADM), sh(C), filesystem(F)

Standards Conformance

volcopy is conformant with:
AT&T SVID Issue 2, Select Code 307-127.

wall

writes to all users

Syntax

/etc/wall

Description

wall reads a message from the standard input until an end-of-file. It then sends this message to all users currently logged in preceded by "Broadcast Message from ...". *wall* is used to warn all users, for example, prior to shutting down the system.

The sender should be super-user to override any protections the users may have invoked.

Files

/dev/tty*

See Also

mesg(C), write(C)

Diagnostics

Cannot send to ...	The open on a user's tty file has failed.

wtinit

object downloader for the 5620 DMD terminal

Syntax

/usr/lib/layersys/wtinit [-d] [-p] file

Description

The *wtinit* utility downloads the named *file* for execution in the AT&T TELETYPE 5620 DMD terminal connected to its standard output. *file* must be a DMD object file. *wtinit* performs all necessary bootstrap and protocol procedures.

There are two options:

-d Prints out the sizes of the text, data, and bss portions of the downloaded *file* on standard error.

-p Prints the downloading protocol statistics and a trace on standard error.

The environment variable **JPATH** is the analog of the shell's **PATH** variable to define a set of directories in which to search for *file*.

If the environment variable **DMDLOAD** has the value **hex**, *wtinit* will use a hexadecimal download protocol that uses only printable characters.

Terminal Feature Packages for specific versions of AT&T windowing terminals will include terminal-specific versions of *wtinit* under those installation sub-directories. */usr/lib/layersys/wtinit* is used for *layers*(C) initialization only when no Terminal Feature Package is in use.

Diagnostics

Returns **0** upon successful completion, **1** otherwise.

Notes

Standard error should be redirected when using the **-d** or **-p** options.

See Also

layers(C)

xbackup

performs XENIX incremental filesystem backup

Syntax

xbackup [key [arguments] filesystem]

Description

xbackup copies all files changed after a certain date in the date in the *filesystem*. *xbackup* is used for XENIX filesystems; use *backup*(ADM) for UNIX filesystems. The *key* specifies the date and other options about the xbackup, where a *key* consists of characters from the set **0123456789kfusd.** The meanings of these characters are described below:

f Places the xbackup on the next *argument* file instead of the default device.

u If the xbackup completes successfully, writes the date of the beginning of the xbackup to the file **/etc/ddate.** This file records a separate date for each file system and each xbackup level.

0-9 This number is the "xbackup level". Backs up all files modified since the last date stored in the file **/etc/ddate** for the same file system at lesser levels. If no date is determined by the level, the beginning of time is assumed; thus the option **0** causes the entire file system to be backed up.

s For xbackups to magnetic tape, the size of the tape is specified in feet. The number of feet is taken from the next *argument*. When the specified size is reached, *xbackup* will wait for reels to be changed. The default size is 2,300 feet.

d For xbackups to magnetic tape, the density of the tape, expressed in BPI, is taken from the next *argument*. This is used in calculating the amount of tape used per write. The default is 1600.

k This option is used when backing up to a block-structured device, such as a floppy disk. The size (in K-bytes) of the volume being written is taken from the next *argument.* If the **k** argument is specified, any **s** and **d** arguments are ignored. The default is to use s and **d.**

If no arguments are given, the *key* is assumed to be **9u** and a default file system is backed up to the default device.

The first xbackup should be a full level-0 xbackup:

xbackup 0u

Next, periodic level 9 xbackups should be made on an exponential progression of tapes or floppies:

xbackup 9u

This progression is shown as follows:

1 2 1 3 1 2 1 4 ...

where xbackup 1 is used every other time, xbackup 2 every fourth, xbackup 3 every eighth, etc.) When the level-9 incremental xbackup becomes unmanageable because a tape is full or too many floppies are required, a level-1 xbackup should be made:

xbackup 1u

After this, the exponential series should progress as if uninterrupted. These level-9 xbackups are based on the level-1 xbackup, which is based on the level-0 full xbackup. This progression of levels of xbackups can be carried as far as desired.

The default file system and the xbackup device depend on the settings of the variables DISK and TAPE, respectively, in the file **/etc/default/backup.**

Files

/etc/ddate	Records xbackup dates of file system/level
/etc/default/backup	Default xbackup information

See Also

cpio(C), default(F), xdumpdir(ADM), xrestore(ADM), restore(ADM), sddate(C), backup(ADM), xbackup(F), *System Administrator's Guide*

Diagnostics

If the xbackup requires more than one volume (where a volume is likely to be a floppy disk or tape), you will be asked to change volumes. Press RETURN after changing volumes.

Notes

Sizes are based on 1600 BPI for blocked tape; the raw magnetic tape device has to be used to approach these densities. Write errors to the xbackup device are usually fatal. Read errors on the file system are ignored.

If the default archive medium specified in */etc/default/xbackup* or */etc/default/restor* is block structured, (i.e. floppy disk) then the volume size in Kbytes must be specified on the command line. Neither utility works correctly without this information. For example, using the default device (below) with the **xbackup** command, enter the following:

xbackup k 360

The default device entry for */etc/default/xbackup* (tape=/dev/xxx) and */etc/default/restor* (archive=/dev/xxx) is */dev/rfd02*.

It is not possible to successfully *restore* an entire active root file system.

Warning

When backing up to floppy disks, be sure to have enough *formatted* floppies ready before starting a xbackup. You must also be sure to close the floppy door when inserting floppy disks. If you fail to do so in a multi-floppy xbackup, the entire xbackup will fail and you will have to begin again.

You should never xbackup more than one filesystem to the tape devices */dev/nrct0* and */dev/nrct2*. This is because, although *xbackup* can write more than one filesystem to */dev/nrct0* or */dev/nrct2*, *restore* may not be able to restore more than one filesystem from these devices.

Value Added

xbackup is an extension of AT&T System V provided in Altos UNIX System V.

xdumpdir

prints the names of files on a XENIX backup archive

Syntax

xdumpdir [**f** filename]

Description

xdumpdir is used to list the names and inode numbers of all files and directories on an archive written with the *xbackup* command. This is most useful when attempting to determine the location of a particular file in a set of backup archives.

The f option causes *filename* to be used as the name of the backup device instead of the default. The backup device depends on the setting of the variable TAPE in the file **/etc/default/xdumpdir**. The device specified as TAPE can be any type of backup device supported by the system (for example, a floppy drive or cartridge tape drive).

Files

rst* Temporary files

See Also

xbackup(ADM), xrestore(ADM), default(F)

Value Added

xdumpdir is an extension of AT&T System V provided in Altos UNIX System V.

xinstall

XENIX installation shell script

Syntax

/etc/xinstall [device]

Description

/etc/xinstall is the *sh*(C) script used to install XENIX distribution (or application program) floppies. It performs the following tasks:

- Prompts for insertion of floppies.
- Extracts files using the *tar*(C) utility.
- Executes **/once/init.*** programs on each floppy after they have been extracted.
- Removes any **/once/init.*** programs when the installation is finished.

The optional argument to the command specifies the device used. The default device is **/dev/xinstall** and is normally linked to **/dev/rdsk/f0q15dt.**

Files

/etc/xinstall

/once/init.*

See Also

custom(ADM), fixperm(ADM), installpkg(ADM)

Notes

xinstall is provided for use with any existing XENIX packages you may have that you wish to install on a UNIX system. *xinstall* does not work with UNIX system applications [use *installpkg*(ADM) to install UNIX system applications].

Value Added

xinstall is an extension of AT&T System V provided in Altos UNIX System V.

xprsetup

transparent printer setup utility

Syntax

xprsetup [**-a**]
xprsetup -d *xprnum*
xprsetup -s *xprnum*

Description

The *xprsetup* utility sets the transparent print mapping for entries listed in the **xprtab**(F) file. Entries may be made in the **xprtab** file using *pcu*(ADM). *xprsetup* is called automatically on transition from single user to multi-user state. Also, *pcu* calls *xprsetup* whenever it makes a change to the system's transparent printer setup.

The first form of the command performs a kernel transparent print map setup for all entries in the **xprtab** file unless the **-a** option is given (see below).

Options

The options to *xprsetup* are:

-a With this option, *xprsetup* displays the transparent print map currently in effect in the kernel for all transparent printer nodes. Transparent printer node pathnames are of the form **/dev/xpr/xpr***XX* where *XX* is in the range 01 through 99.

-d With this option, *xprsetup* displays the transparent print map currently in effect in the kernel for transparent printer node *xprnum*. The argument *xprnum* must be a decimal number in the range 1 through 99.

-s This option performs a kernel transparent print map setup for transparent printer node *xprnum* as specified in the **xprtab** file. If no entry exists for *xprnum* in the **xprtab** file then any current kernel transparent print map setup for transparent print node *xprnum* is canceled. The argument *xprnum* must be a decimal number in the range 1 through 99.

Files

/etc/xprtab
/dev/xpr/xpr??
/etc/rc.d/6/xprinit

See Also

pcu(ADM), xprtab(F), xprcat(C)

Value Added

xprsetup is an extension to AT&T UNIX System V provided in Altos UNIX System V.

xrestore, xrestor

invokes XENIX incremental filesystem restorer

Syntax

xrestore key [arguments]

xrestor key [arguments]

Description

xrestore is used to read archive media backed up with the *xbackup*(ADM) command.

The *key* specifies what is to be done. *Key* is one of the characters **cC**, **rR**, **tT**, or **xX** optionally combined with **k** and/or **f** or **F**. *restor* is an alternate spelling for the same command.

c,C
Verify (check) a dump tape. Used after a dump is made to make sure the tape has no I/O errors or bad checksums. **C** is the same as **c** except that it provides a higher level of checking.

f Uses the first *argument* as the name of the archive (backup device /dev/*) instead of the default.

F **F** is the number of the first file on the tape to read. All files up to that point are skipped.

k Follow this option with the size of the backup volume. This allows for reading multivolume dumps from media such as floppies.

r,R
The archive is read and loaded into the file system specified in *argument*. This should not be done lightly (see below). If the key is **R**, *xrestore* asks which archive of a multivolume set to start on. This allows *xrestore* to be interrupted and then restarted (an *fsck* must be done before the restart).

t Prints the date the archive was written and the date the file system was backed up.

T Prints a full listing of a dump tape. Similar to **t**.

x Each file on the archive named by an *argument* is extracted. The filename has all "mount" prefixes removed; for example, if **/usr** is a mounted file system, **/usr/bin/lpr** is named **/bin/lpr** on the

archive.
The extracted file is placed in a file with a numeric name supplied by *xrestore* (actually the inode number). In order to keep the amount of archive read to a minimum, the following procedure is recommended:

1. Mount volume 1 of the set of backup archives.

2. Type the *xrestore* command with the appropriate key and arguments.

3. *xrestore* will check *xdumpdir*, then announce whether or not it found the files, give the numeric name that it will assign to the file, and in the case of a tape, rewind to the start of the archive.

4. It then asks you to "mount the desired tape volume". Type the number of the volume you choose. On a multivolume backup, the recommended procedure is to mount the last through the first volumes, in that order. *xrestore* checks to see if any of the requested files are on the mounted archive (or a later archive, thus the reverse order). If the requested files are not there, *xrestore* doesn't read through the tape. If you are working with a single-volume backup or if the number of files being xrestored is large, respond to the query with 1 and *xrestore* will read the archives in sequential order.

X Same as **x** except that files are replaced in original location. When you use this option, omit the initial slash (/) in the filename on the *xrestore* command line.

The **r** option should only be used to xrestore a complete backup archive onto a clear file system, or to xrestore an incremental backup archive onto a file system so created. It should not be used to xrestore a backup archive onto the root file system. Thus:

```
/etc/mkfs /dev/hd1 10000
xrestore r /dev/hd1
```

is a typical sequence to xrestore a complete backup. Another *xrestore* can be done to get an incremental backup in on top of this.

A *xbackup* followed by a *mkfs* and a *xrestore* is used to change the size of a file system.

Files

rst*	Temporary files
/etc/default/restor	Name of default archive device

The default archive unit varies with installation.

Notes

It is not possible to successfully *xrestore* an entire active root file system.

Note also that *xrestore* may be unable to xrestore more than one filesystem from the tape devices */dev/nrct0* and */dev/nrct2*.

Diagnostics

There are various diagnostics involved with reading the archive and writing the disk. There are also diagnostics if the i-list or the free list of the file system is not large enough to hold the dump.

If the dump extends over more than one disk or tape, *xrestore* may ask you to change disks or tapes. Reply with a newline when the next unit has been mounted.

See Also

xbackup(ADM), xdumpdir(ADM), fsck(ADM), mkfs(ADM), sddate(C)

Value Added

xrestor and *xrestore* are extensions of AT&T System V provided in Altos UNIX System V.

xts

extract and print xt driver statistics

Syntax

xts [**-f**]

Description

The *xts* command is a debugging tool for the *xt*(HW) driver. It performs an **XTIOCSTATS** *ioctl*(S) call on its standard input file to extract the accumulated statistics for the attached group of channels. This call will fail if statistics have not been configured in the driver, or the standard input is not attached to an *xt*(HW) channel. The statistics are printed one item per line on the standard output.

-f Causes a ''formfeed'' character to be put out at the end of the output for the benefit of page-display programs.

Diagnostics

Returns **0** upon successful completion; **1** otherwise.

See Also

xtd(ADM), xtt(ADM), xt(HW), ioctl(S), xtproto(M)

xtt

extract and print xt driver packet traces

Syntax

xtt [-f] [-o]

Description

The *xtt* command is a debugging tool for the *xt*(HW) driver. It performs an **XTIOCTRACE** *ioctl*(S) call on its standard input file to turn on tracing and extract the circular packet trace buffer for the attached group of channels. This call will fail if tracing has not been configured in the driver, or the standard input is not attached to an *xt*(HW) channel. The packets are printed on the standard output.

The optional flags are:

-f Causes a "formfeed" character to be put out at the end of the output for the benefit of page-display programs.

-o Turns off further driver tracing.

Diagnostics

Returns **0** upon successful completion; **1** otherwise.

Note

If driver tracing has not been turned on for the terminal session by invoking *layers*(C) with the **-t** option, *xtt* will not generate any output the first time it is executed.

See Also

layers(C), xtd(ADM), xts(ADM), xt(HW), ioctl(S), layers(M)

Altos UNIX® System V/386
Release 3.2

(HW) Hardware

Contents

Hardware Dependent **(HW)**

Intro

introduction to machine related miscellaneous features and files

Description

The hardware-dependent section (HW) contains information useful in maintaining the system. Included are descriptions of files, devices, tables and programs that are important in maintaining the entire system.

audit

audit subsystem interface device

Description

The audit subsystem provides two minor devices for interfacing to the audit subsystem. One device, **/dev/auditr,** is used exclusively by the audit daemon, *auditd*(ADM), for the purpose of reading the subsystem audit collection file records. The other device, **/dev/auditw,** is used by application programs which are privileged to write audit records to the audit subsystem. This device may be opened by as many applications as is necessary but may only be opened for writing. The device also support a host of *ioctl*(S) functions to perform audit subsystem control.

The audit read device provides the usual character device driver *open*(S), *read*(S), and *close*(S) routines. Writing to this device is not permitted. Read requests are satisfied by the subsystem and optimize the efficiency of the daemon and the performance of the system. Read requests are satisfied when sufficient data has accumulated to meet an administrator specified threshold. Until the data is available, the read request will block. In this manner, the daemon will receive sufficiently large blocks of data on each read to allow sufficient compaction. Also, context switch frequency is greatly reduced since the reads will not be satisfied on small blocks or when no data is available.

The audit write device provides an interface to the audit subsystem for applications that have the **writeaudit** privilege. The device supports the *openR(S)*, *close*(S), *write*(S), and *ioctl*(S) entry points. Once opened, an application may compose an audit record and *write*(S) it to the device for inclusion in the collection file. The writing of an audit record is an atomic action in that the entire record must be presented to the subsystem with a single write. It is incumbent on the application to gather the record into a single buffer before writing it to the device.

The format of an audit record depends upon the type of event being audited. All audit records begin with a common audit record header defined by the audit_header structure in the file **sysaudit.h**.

```
struct audit_header {
     ushort     rec_length;     /* total record length */
     time_t     tstamp;         /* date/time of record */
     ulong      event_id;       /* event sequence id */
     ushort     event_type;     /* event classification */
     ushort     record_type;    /* record format */
     ushort     obj_type;       /* object type */
     ushort     pid;            /* process_id */
};
```

The *event_type*, *record_type*, and *pid* fields must be filled by the application; all other fields are filled by the Audit subsystem. The event types are defined in the header file and provide a method of categorizing audit records into groups such as Login events or System Administrator events. The record type informs the subsystem of the record template type. This information is also retained with the record when it is written to the collection file by the subsystem since it is required at data reduction time.

Some of the record types have variable length string areas that follow the fixed portion of the audit record. Each text string that is part of the record has its size recorded in a count field. Each string is null-terminated and the count must include the null character. When the record is written to the device, the amount of data written includes the fixed portion plus all text strings. The supported record types for application programs are:

RT_LOGIN	login/logoff events
RT_PASSWORD	password modifications
RT_DATABASE	protected database modifications
RT_SUBSYSTEM	privileged subsystem events
RT_LOCK	terminal and account locking
RT_AUDIT	audit subsystem events

Each record type indicates a unique record structure definition. The **RT_LOGIN** record uses the login_audit structure. It contains the following fields, defined in **sys/audit.h**:

```
struct login_audit {
        struct audit_header aud_hdr;
        char    username[8];     /* login name */
        ushort  code;            /* function code */
        ushort  luid;            /* login userid */
        ushort  rgid;            /* real gid */
        dev_t   ttyd;            /* controlling terminal */
        ptr_t   cdir;            /* current directory */
        ptr_t   terminal;        /* stdin terminal name */
#ifdef B1
        ptr_t   sec_level;       /* login sensitivity level */
#endif
};
```

Username is the login or logoff user account name. The luid and rgid fields are those associated with the specified user account. The audit header, which precedes the login specific portion of the record, must have the *record_type* field set to **RT_LOGIN**. The *event_type* used for login/logoff is the **ET_LOGIN** event.

The *code* field is used to distinguish between specific actions that may fall into a common category. For instance, the **ET_LOGIN** event category includes both successful and unsuccessful logins, and also logoffs. The code values, defined in the header file, indicate which of these occurred.

The login audit record also contains two variable length text strings. These are the login terminal and the process current directory. The string area begins immediately following the fixed portion of the record. The size of each text string field is indicated by the ptr_t typedef field which contains the length of the string including the null character. The null character is considered part of the string. Once the strings have been calculated and the record completed, the length field in the audit record header is set to the size of structure plus the total lengths of the strings. This is the amount of data to *write*(S) to the audit device.

Modifications to user passwords are audited by the password management subsystem. Each attempt, whether successful or not, results in an audit record of type **RT_PASSWORD** being generated. The structure is defined in the *sys/sudit.h* header file:

```
struct passwd_audit {
        struct audit_header aud_hdr;
        char    username[8];   /* login user name */
        ushort code;           /* function code */
};
```

The code value distinguishes between successful and unsuccessful attempts to change the password on the indicated user account.

The system maintains a number of protected database files to support the system security policy. Attempts to modify the databases are audited with the **RT_DATABASE** type records. These records have the following format, as defined in **<sys/audit.h>**:

```
struct database_activity {
    struct audit_header aud_hdr;
    ptr_t     command;        /* command name */
    ushort    code;           /* Type of database audit */
    ushort    object;         /* object type */
    long      expected_val;   /* Expected value of parameter */
    long      present_val;    /* Present value of parameter */
    ptr_t     action;         /* security action that failed */
    ptr_t     result;         /* result of failure */
};
```

The *dbase* and *code* values identify the database and the specific action whether successful or not. A variable length text string area is provided to identify precisely what database *field* along with the *old* and *new* database field values. The audit header *length* field includes the size of the string area and the fixed portion of the record.

Protected subsystems use the **RT_SUBSYSTEM** record type to record security related events that occur in subsystem components. *Code* is used to identify the subsystem generating the record. Both the command and resulting action as well as the resulting failure are recorded in *command, action* and *result* respectively.

```
struct subsystem_activity {
    struct audit_header aud_hdr;
    ptr_t      command;     /* command name */
    ushort     code;        /* Subsystem type */
    ptr_t      action;      /* action that failed */
    ptr_t      result;      /* result of failure */
};
```

The **RT_LOCK** record type is used to audit user account and terminal locking events. The *username* identifies the user account which was locked or unlocked. *Code* distinguishes between the several events that result in the generation of a lock audit record.

```
#include<sys/audit.h>

struct lock_audit {
        struct audit_header aud_hdr;
        char    username[12];    /* login username */
        ushort  code;            /* lock function code */
        ushort  trys;            /* failed attempts */
};
```

Programs that interact with and control the audit subsystem are audited with the **RT_AUDIT** record type. The subsystem is enabled and disabled by an application program. The same is true of subsystem parameter initialization and modification. Events such as the initiation and termination of the audit daemon, the execution of the recovery mechanism, data reduction and report generation, and audit file archival are all audited.

The text string portion of the audit record is only applicable for the audit enable function since the initial subsystem collection file must be specified for the daemon log file. All other audit records do not use this field. The *code* indicates which of the above events took place.

```
struct audit_actions {
    struct audit_header aud_hdr;
    ushort     code;     /* audit function code */
    ptr_t      text1;    /* initial collection file */
};
     struct passwd_audit {
             struct audit_header aud_hdr;
             char    username[8];  /* login user name */
             ushort code;             /* function code */
     };
```

The audit device supports a number of *ioctl(S)* functions to control the audit subsystem. The format of the *ioctl(S)* calls is:

ioctl (fildes, command, arg)
int fildes, command;
struct audit_init *arg;
-or-
struct audit_ioctl *arg;
-or-
struct audit_stats *arg;

The audit_init structure is only used for **AUDIT_ENABLE** command to perform subsystem initialization. The structure is defined as follows:

```
struct audit_init {
     uint      buf_length;      /* lngth of data including header */
     mask_t    audit_flags[1];  /* audit control flags */
     mask_t    event_mask[AUDIT_MASK_SIZE];     /* system event mask */
     uint      read_count;      /* daemon read count to satisfy */
     uint      write_count;     /* write count for coll. file flush */
     long      write_time;      /* write flush time in seconds */
     long      switch_count;    /* collection file size maximum */
     long      caf_maxsize;     /* compacted audit file max size */
     uint      dir_count;       /* directory count */
     uint      uid_count;       /* uid selection count */
     uint      gid_count;       /* gid selection count */
     ulong     dir_offset;      /* fseek of directory names */
     ulong     uid_offset;      /* fseek of uids to select */
     ulong     gid_offset;      /* fseek of gids to select */
     uint      buff_count;      /* number of collection file buffers */
     ulong     session;         /* system boot session number */
     short     audit_uid;       /* audit user uid */
     short     audit_gid;       /* audit group gid */
};
```

The subsystem initialization parameters are established through the menu interface and are written to a parameter file. This file is read and used to fill out the above structure to initialize the subsystem.

The *event_mask* is a bit mask of the selected events to audit during the session. Only events that are enabled will generate audit records. The *read_count* value is used by the subsystem to satisfy audit daemon reads. Only when the specified amount of data is available in the collection file will the read be satisfied.

The flushing of the internal subsystem buffers to the collection file is controlled by the *write_count* and *write_time* fields. When the specified amount of data has accumulated, the buffers will be flushed to disk. A time interval in seconds can also be set which will cause the flushing of data to disk after a certain period of elapsed time.

The *switch_count* controls the size to which subsystem collection files may grow until a file switch is performed. The size of the output compaction files written by the audit daemon are controlled by the *caf_maxsize* parameter. When these files reach this specified size, the daemon performs a switch to a new compaction file and records this fact in the audit session log file. *Session* is the current session value that is used in file name generation. The *buff_count* value determines the number of file system blocksize buffers to be allocated by the subsystem for the purpose of internal buffering. At least 2 buffers are allocated while 4-6 is optimal.

Dir_count is the number of collection file and compaction file directories that are available to both the subsystem and the audit daemon for the creation of their respective files. If a file write error occurs, both will attempt to use an alternate directory. Both will terminate only when all directories have been tried without success. The directory names are located in the variable length directory area following the fixed portion of the initialization record. Each pathname is a null-terminated string. The *dir_offset* field points to the start of this variable length text string area with respect to the start of the structure.

The audit subsystem is capable of selective audit record generation based on user and group IDs. These values may be specified to the subsystem at initialization time using the *uid_count* and *gid_count* values. The actual list of user and group Ids are located at the end of the structure in a variable length table of short integers. The offsets where the ID arrays may be found are located by the *uid_offset* and *gid_offset* values.

The *audit_uid* and *audit_gid* fields are used to communicate certain ID values to the subsystem since these are used to create files with specific owners and groups for security purposes.

All remaining *ioctl(S)* commands except AUDIT_STATS use the audit_ioctl structure. The audit_ioctl structure is defined by the following:

```
struct audit_ioctl {
     uint       read_count;      /* daemon read count */
     uint       write_count;     /* write count for file flush */
     long       write_time;      /* write flush time */
     mask_t     user_control[AUDIT_MASK_SIZE]; /* control mask */
     mask_t     user_disp[AUDIT_MASK_SIZE];    /* disposition mask */
     mask_t     system_mask[AUDIT_MASK_SIZE];  /* system event mask */
};
```

The AUDIT_STATS ioctl command uses the following structure for statistic retrieval and display.

```
struct audit_stats {
     uint     session;         /* current session number */
     uint     sequence;        /* current sequence number */
     ulong    total_bytes;     /* total bytes written */
     ulong    total_recs;      /* total records written */
     ulong    syscall_recs;    /* system call audit record count */
     ulong    syscall_norecs;  /* system call audit record count */
     ulong    appl_recs;       /* application audit record count */
```

```
        ulong      read_count;      /* number of device reads */
        ulong      write_count;     /* number of device writes */
        ulong      coll_files;      /* number of collection files */
        ulong      buffers_used;    /* maximum audit buffer usage */
        ulong      buffer_sleep;    /* number of audit write sleeps */
};
```

The commands supported by the audit device are:

ENABLE
: Initialize and enable the audit subsystem for the generation of audit records.

SHUTDOWN
: Notify the audit subsystem that a system shutdown is in progress.

DISABLE
: Terminate the audit subsystem and close all collection files. The audit daemon is also terminated after the last audit record has been read from the subsystem.

SYSMASK
: Modify the audit subsystem event mask that controls the generation of audit records based on certain event types.

USERMASK
: Modify the user event mask for a process. Each process has a mask which can be used to always or never audit certain event types regardless of the system event mask. The mask is a control mask which indicates for each bit set on that the generation of records for the corresponding event type is controlled by the second mask. The second mask is the enable/disable mask which determines whether the event is always or never audited. If a control mask bit is 0, the event is controlled by the system event mask.

FLUSH
: Modify the write count and time interval values.

DAEMON
: Modify the audit daemon read count value.

ACK
: Used by the daemon to acknowledge certain events such as recognition of system shutdown and the disabling of the audit subsystem. Provides a synchronization means between the subsystem and the daemon.

MOUNT
: The system has transitioned to multi-user state and alternate audit directories are now mounted and available.

STATS
: Retrieve the current audit subsystem statistics from the audit device.

IDS Specify the user and group IDs to use for selective audit generation.

Ioctl(S) calls will fail if any of the following are true:

[EPERM] The process required SelfAudit privilege but did not have it.

[EEXIST] An attempt is made to enable audit and it is already running.

[EACCES] An open attempt is made on the audit device and the calling process does not have the **configaudit** or **writeaudit** authorization.

[EBADF] *Fildes* is not a valid open file descriptor.

[EFAULT] *Arg* points to an illegal address.

[EINVAL] *Command* is an illegal value.

Files

/dev/auditr
/dev/auditw

See Also

auditd(ADM), auditcmd(ADM), "Maintaining System Security," chapter of the *System Administrator's Guide*

Diagnostics

Upon successful completion, the device returns a 0. Otherwise, a -1 is returned and *errno* is set to indicate the error.

Value Added

audit is an extension of AT&T System V provided in Altos UNIX System V.

boot

UNIX boot program

Description

boot is an interactive program used to load and execute stand-alone UNIX programs. It is used primarily for loading and executing the Altos UNIX System V kernel, but can load and execute any other programs that are linked for stand-alone execution. *boot* is a required part of the Operating System and must be present in the root directory of the root filesystem to ensure successful loading of the Altos UNIX System V kernel.

The *boot* program is invoked by the system each time the computer is started. To restart the system without going through lengthy shutdown procedures, you can use the *reboot* command. This causes the system to reboot after shutting down without waiting for keyboard input. See *haltsys*(ADM) for more information.

For diskette boot, the procedure has three stages:

1. The ROMs load the boot block from sector 0 of the floppy, where sector 0 of the disk is the same as sector 0 of the filesystem.

2. The boot block loads *boot* from the floppy filesystem.

3. *boot* executes and prompts the user.

For fixed-disk boot, the procedure has five stages:

1. The ROMs load in the *masterboot* block from sector 0 on the hard disk.

2. The *masterboot* block then loads the partition boot block (boot0) from sector 0 of the active partition (see *fdisk*(ADM)).

3. Then, assuming the UNIX partition is active, boot1 is loaded from 1K into the active partition. Boot1 spans 20 physically contiguous 1K blocks on the disk.

4. boot1 loads *boot* from the UNIX filesystem.

5. *boot* executes and prompts the user.

The fixed-disk boot procedure is invoked if the diskette drive is empty.

When first invoked, *boot* prompts for the location of a program to load by displaying the message:

```
Altos UNIX System V/386

Boot
:
```

To specify the location of a program, a device and filename must be given. The filename must include the full pathname of the file containing the stand-alone program. You can display a list of the current allowable device names by typing a question mark (?).

The format for the device and pathname is as follows:

xx(*a*,*b*,*c*,*p*,*d*)*filename*

where:

The *a*, *b*, *c*, *p*, and *d* parameters are each a number.
The *b*, *c*, and *p*, parameters are optional.

xx = device name
('hd' for the hard disk or 'fd' for diskette device)
filename = standard UNIX pathname. Must start with a slash if the program is not in the root directory.

The meaning of parameters *a* and *b* depends upon the number parameters used, as described below:

xx(*a*)*filename*

a = minor device number of disk device.

xx(*a*,*b*)*filename*

a = minor device number of disk device.
b = offset from start of disk.

xx(*a*,*b*,*c*)*filename*

a = EISA slot number of disk device
b = SCSI channel (host adapter) number of disk device.
c = SCSI ID number of disk device.

In this form, the default disk partition number is 5 (the active partition), and the default disk division is 0 (zero).

xx(a,b,c)filename

a = EISA slot number of disk device
b = SCSI channel (host adapter) number of disk device.
c = SCSI ID number of disk device.
p = disk partition number

xx(a,b,c)filename

a = EISA slot number of disk device
b = SCSI channel (host adapter) number of disk device.
c = SCSI ID number of disk device.
p = disk partition number
d = disk division number

All numbers are in decimal. See the manual pages for *hd*(HW) and *fd*(HW) for minor device numbers of these devices. The location of the program to be loaded must always be entered first on the command line and be present if other boot options are specified either on the command line or in **/etc/default/boot**. Alternatively, the *xx(a,b,c,p,d)* portion of the string can be omitted altogether, and only the *filename* is used. In this form, an internal default is used for all parameters not specified. The specific default parameters are site-dependent.

If you want *boot* to pause and wait for a RETURN before executing the program that it loads, enter the word "prompt" on the command line. For example, if you enter "prompt" and press RETURN *boot* prints the following message and waits for you to press the RETURN key again:

```
Loaded, press <RETURN>.
```

The prompt can be changed to another string as in this example:

```
prompt="change diskettes now"
```

boot loads **unix** from the diskette, prints the message "change diskettes now", and waits for RETURN to be pressed. No other characters can appear between "prompt", the "=" sign, and the prompt string, although *string* may contain spaces. When you press RETURN **unix** begins execution. "Prompt" can be set either on the command line or in **/etc/default/boot**. If a prompt is not specified, *boot* executes the loaded program without pausing.

If you have just loaded the *boot* program from the distribution diskette, simply press <RETURN> and *boot* defaults to the correct values.

To load from a hard disk with an ID 0 that is attached to the first channel (0) of the board in the first slot (1), enter:

```
hd(1,0,0)unix
```

To use the default bootstring specified in **/etc/default/boot**, simply press <RETURN> when the system displays the boot prompt, and *boot* uses the values specified by DEFBOOTSTR in **/etc/default/boot**.

If nothing is typed after a short while and AUTOBOOT is set to YES in the default *root* filesystem's **/etc/default/boot** file, *boot* times out and behaves as though a <RETURN> had been pressed, except that an "auto" is added to the boot string. (If, in addition to AUTOBOOT=YES, TIMEOUT=*n* is defined, *boot* waits *n* seconds before timing out.) *boot* proceeds through the boot procedure, and *init*(M) is passed a **-a** flag with no "prompt".

If you wish to install DOS on the hard disk, it is recommended that you do so before you install the Operating System. See the manual page for *dos*(C). However, once you install DOS, you can boot it at the UNIX boot prompt by entering "dos".

During installation, a custom *masterboot* is placed on the hard disk. If a non-standard disk is specified, its parameters are stored and enabled in this *masterboot*.

Configuring the Kernel

boot passes portions of the bootstring typed at the boot prompt to the kernel.

The kernel reads the bootstring to determine which peripherals are the root, pipe, and swap devices. If no devices are specified in either the **/etc/default/boot** description or on the command line, the default devices compiled into the kernel are used.

Additional arguments in the bootstring can alter this default action. These arguments have the form:

dev=*xx*(*a*,*b*)

where:
dev = the desired system device (e.g., **root**, **pipe**, or **swap**)
xx = the desired boot device (e.g., **hd** or **fd**)

If only the *a* parameter is used, in this form:

dev=*xx*(*a*)

then:
a = minor device number

If the (*a*,*b*) pair is used, as in this form:

dev=xx(*a*,*b*)

then:

a = major device number
b = minor device number

If any combination of **root**, **pipe**, or **swap** is specified, then those system devices will reside on that device, with the unspecified system devices using the defaults compiled in the kernel. Setting one device does not affect the default values for the other system devices.

Selecting the System Console

You can select the system console at boot time either by entering the command **systty**=*x* at the boot prompt, or by placing the keyword **SYSTTY**=*x* in the file **/etc/default/boot**. The letter *x* represents either a number or a string parameter.

If you use the **systty**=*x* command at boot time, *boot* uses the string parameter *x* to pass the selected console device to the kernel. The values of the bootstring parameter **systty** are:

sio	Serial port COM1
cn	Display adapter

For example, to assign the system console to the serial port at COM1, enter this command at the boot prompt:

```
systty=sio
```

If you do not specifically set the system console at boot time, the *boot* program follows these steps to determine the system console:

- *boot* reads **/etc/default/boot** and looks for the keyword **SYSTTY**=*x*, where *x* is a number that specifies the system console device.

  1 indicates the serial adapter at COM1
  0 indicates the display adapter

- If **SYSTTY** is not found or **/etc/default/boot** is unreadable, *boot* checks for a display adapter and assigns it as the system console.

- If no display adapter is found, *boot* looks for COM1, sets the serial port to 9600 baud, 8 data bits, 1 stop bit, and no parity, and uses it as the system console.

Thus, to have *boot* automatically set the system console to the serial port at COM1, enter this line in **/etc/default/boot**:

SYSTTY=1

Aliasing

A set of system devices can be aliased to a single keyword by defining the keyword in the file **/etc/default/boot**. This keyword can then be entered on the "Boot" command line and the *boot* program then reads the corresponding system devices from **/etc/default/boot** and pass them to the kernel. An alias has the following form:

key=file [root=*xx*(*a,b,c,p,d*) pipe=*xx*(*a,b*) swap=*xx*(*a,b*) prompt[="*string*"]]

The following is an example alias string that can be added to the **/etc/default/boot** file:

harddisk=hd(1,0,0)unix root=hd(40) pipe=hd(40) swap=hd(41)

The next time you boot the system, enter **harddisk** in response to the "Boot:" prompt. The effect will be equivalent to having typed the string that **harddisk** is aliased to.

Other Command Line Parameters

Several other command line parameters are recognized by *boot*, as summarized below:

verbose=no
verbose=yes

Whether or not to display initialization messages during system boot. The default action is specified in the file **/etc/default/boot,** but the actual mode can be modified on the command line. See the "Boot Options" section below.

restart=no
restart=yes

Whether to attempt a UPS shutsave restart operation when a valid saved restart image exists on the restart disk device. The default action is specified in the file **/etc/default/boot**, but the actual mode can be modified on the command line. See the "Boot Options" section below.

Boot Options

Boot options can be changed via keywords in **/etc/default/boot**. The following keywords are recognized by *boot*:

AUTOBOOT=YES	If YES, *boot* automatically loads Altos UNIX System V after a delay time specified by the TIMEOUT parameter. The default value is 30 seconds.
DEFBOOTSTR=*string*	*string* is used as the default bootstring for timeouts or when only a <RETURN> is entered at the boot prompt. There can be no white space between DEFBOOTSTR, the "=" sign and *string*.
SYSTTY=*x*	If *x* is **1**, the system console device is set to the serial adapter at COM1. If *x* is **0** the system console is set to the main display adapter.
RONLYROOT=NO	Whether or not the root filesystem is to be mounted *readonly*. This should only be set to YES during installation.
FSCKFIX=YES or NO	Whether or not *fsck*(ADM) fixes any root system problems by itself. If the variable is set to YES, then *fsck* is run on the root filesystem with the **-rr** flag.
MULTIUSER=YES or NO	Whether or not *init*(M) invokes *sulogin* or proceeds to multiuser mode.
PANICBOOT=YES or NO	Whether or not the system reboots after a panic(). This variable is read from **/etc/default/boot** by *init*.
TIMEOUT=*n*	*n* is the number of seconds to wait at the boot prompt before timing out and booting the kernel (if AUTOBOOT is set to YES).
VERBOSE=YES or NO	Whether or not to display initialization messages during boot. Default is NO, which causes kernel messages to be logged in **/usr/adm/messages**, and startup script messages to be logged in **/etc/rclog**.

RESTART=YES or NO
: Whether to attempt a UPS shutsave restart operation (see *upsconfig*(ADM)). If set to **YES**, a restart will always happen after a shutsave operation had brought the system down. If set to **NO**, the system will always perform a normal bootstrap.

RSPART=*xx*(*a*,*b*,*c*,*p*,*d*)
: Specifies the shutsave restart disk device. The convention used for *xx* and *a*, *b*, *c*, *p*, and *d* are identical to the boot device description, except there is no filename. See *upsconfig*(ADM).

The following special boot options are for intended for use applications with a special need to alter init's tolerance for processes that need to be restarted.

SPAWN_INTERVAL
: The number of seconds over which "init" will try to respawn a process SPAWN_LIMIT times before it gets mad. The default value is 120.

SPAWN_LIMIT
: The number of respawns "init" will attempt in SPANW_INTERVAL seconds it generates an error message and inhibits further tries for INHIBIT seconds. The default value is 10.

SLEEPTIME
: Sets the time (in seconds) between calls to *sync*.

INHIBIT
: The number of seconds "init" ignores an entry it had trouble spawning unless a "telinit Q" is received. The default value is 300.

Diagnostics

If an error occurs, *masterboot* displays an error message, and locks the system. The following is a list of the most common messages and their meanings:

IO ERR An error occurred when *masterboot* tried to read in the partition boot of the active operating system.

BAD TBL The bootable partition indicator of at least one of the operating systems in the *fdisk* table contains an unrecognizable code.

NO OS There was an unrecoverable error that prevented the active operating system's partition boot from executing.

When *boot* displays error messages, it returns to the "Boot" prompt. The following is a list of the most common messages and their meanings:

bad magic number
The given file is not an executable program.

can't open *<pathname>*
The supplied pathname does not correspond to an existing file, or the device is unknown.

Stage 1 boot failure
The bootstrap loader cannot find or read the **boot** file. You must restart the computer and supply a filesystem disk with the **boot** file in the root directory.

not a directory
The specified area on the device does not contain a valid UNIX filesystem.

zero length directory
Although an otherwise valid filesystem was found, it contains a directory of apparently zero length. This most often occurs when a pre- System V UNIX filesystem (with incorrect, or incompatible word ordering) is in the specified area.

fload:read(*x***)=***y*
An attempted read of *x* bytes of the file returned only *y* bytes. This is probably due to a premature end-of-file. It could also be caused by a corrupted file, or incorrect word ordering in the header.

Files

/boot
/etc/default/boot
/etc/masterboot
/etc/hdboot0
/etc/hdboot1

See Also

autoboot(ADM), badtrk(ADM), fd(HW), fdisk(ADM), fsck(ADM), haltsys(ADM), hd(HW), init(M), sulogin(ADM), upsconfig(ADM)

Notes

The computer tries to boot off any diskette in the drive. If the diskette does not contain a valid bootstrap program, errors occur. The *boot* program can be used to load other standalone programs besides the UNIX kernel, but these standalone programs must be in COFF binary format. (See *a.out*(F).) Moreover, these programs must be linked for standalone execution.

RONLYROOT should only be set to YES for installation. If it is set to YES during day-to-day operations, it will prevent your making changes to the root filesystem. You will then be required to boot from the floppy drive, edit the **/etc/default/boot** file, and reboot.

Value Added

boot is an extension of AT&T System V provided in Altos UNIX System V.

cdrom

compact disk devices

Description

The cdrom devices implement the interface with compact disk drives.

The character special cd devices (**/dev/rcd0**, and so forth) support raw I/O in multiples of the physical sector size of the CD-ROM (typically 2048 bytes).

The block special cd devices (**/dev/cd0** and so forth) support buffered I/O.

The minor device number determines which compact disk unit will be accessed. The correspondence beween the unit number and the SCSI host adaptor, controller and lun is defined in the SCSI configuration file **/etc/conf/cf.d/mscsi.**

Files

/dev/cd[0-n]
/dev/rcd[0-n]
/usr/lib/mkdev/cdrom

See Also

scsi(HW), mkdev(ADM)

Notes

Because the CD-ROM is a read-only device it is only possible to open it for input.

The command *mkdev cdrom* can be used to interactively configure the CD-ROM driver.

cmos

displays and sets the configuration data base

Syntax

cmos [address [value]]

Description

The *cmos* command displays and/or sets the values in the CMOS configuration data base. This battery-powered data base stores configuration information about the computer that is used at power up to define the system hardware configuration and to direct boot procedures. The data base is 64 bytes long and is reserved for system operation. Refer to your computer hardware manual for more information.

The *cmos* command is typically used to alter the current hardware configuration when new devices are added to the system. When only *address* is given, the command displays the value at that address. If both *address* and a *value* are given, the command assigns the value to that address. If no arguments are given, the command displays the entire contents of the data base.

The CMOS configuration data base may also be examined and modified by reading from and writing to **/dev/cmos** file. Because successful system operation depends on correct configuration information, the data base should be modified by experienced system administrators only.

The diagnostic diskette should be run before setting the CMOS data base.

Note that this section refers to the standard AT CMOS, not the EISA nonvolatile memory, which is maintained through the EISA Configuration Utility.

Files

/etc/cmos
/dev/cmos

fd

floppy devices

Description

The **fd** devices implement the interface with floppy disk drives. Each device name corresponds to a specific major and minor device. Typically, the *tar*(C), *cpio*(C) or *dd*(C) commands are used to read or write floppy disks. For instance,

tar tvf /dev/fd0

tabulates the contents of the floppy disk in drive 0 (zero).

The block special **fd** devices are also block-buffered. The floppy driver can read or write 1K bytes at a time using raw i/o. Note that block transfers are always a multiple of the 1K disk block size.

XENIX Devices

XENIX diskette device file names use the following format:

/dev/[r]fd[0,1][48ss8,48ss9,96ds9,96ds15,135ds9,135ds18]

(See Notes, below, for more information about device naming procedure.) The corresponding character special (raw) devices afford direct, unbuffered transmission between the floppy and the user's read or write transfer address in the user's program.

For information about formatting, see *format*(C).

The minor device number determines what kind of physical device is attached to each device file (see Notes). When accessing the character special floppy devices, the user's buffer must begin on a word boundary. The count in a *read*(S), *write*(S), or *lseek*(S) call to a character special floppy device must be a multiple of 1K bytes.

Device names determine the particular drive and media configuration. The device names have the form:

fd048ds9

Where:

fd0 = drive number (0, 1, 2 or 3)
48 = number of disk tracks per inch (48 or 96)
ds = single or double sided floppy (ss or ds)
9 = number of sectors on the floppy (8 or 9)

For instance, /dev/fd048ss9 indicates a 48 track per inch, single sided, 9 sector floppy disk device in drive 0.

The minor device numbers for floppy drives depend on the drive and media configuration. The most common are:

	48tpi				96tpi		135tpi	
	ds/8	ds/9	ss/8	ss/9	ds/15	ds/8	ds/9	ds/18
Drive	Minor Device Number							
0	12	4	8	0	52	44	36	60
1	13	5	9	1	53	45	37	61
2	14	6	10	2	54	46	38	62
3*								

* reserved for special, non-floppy devices connected to the floppy controller as unit #3.

The scheme for creating minor device numbers is as follows. When interpreted as a binary number, each bit of the minor device number represents some aspect of the device/media configuration.

For example, the minor device number for **/dev/fd048ss8** is "8." Interpreted as a binary number, 8 is:

00001000

This is how each bit, or binary digit, is significant:

48tpi - 0 96tpi - 1 135tpi - 1	Sectors per Track		ss - 0 ds - 1	Drive	
32	16	8	4	2	1
0	0	1	0	0	0

Only the last six digits of the number are used in minor device identification. The first significant digit is the third from the left. In this example, the third digit from the left is zero, thus the device is 48tpi. The next two digits mean:

Bits		Sectors per Track
16	8	
0	0	9
0	1	8
1	0	15
1	1	18

The fourth digit tells whether the floppy is single sided (ss - 0) or double sided (ds - 1). The last two signify the drive number:

Bits		Drive Number
2	1	
0	0	0
0	1	1
1	0	2
1	1	3*

* reserved for special, non-floppy devices connected to the floppy controller as unit #3.

Using this information, you can construct any minor device numbers you need.

UNIX Devices

UNIX diskette device file names use the following format:

/dev/[r]dsk/f[0,1][5h,5d9,5d8,5d4,5d16,5q,3h,3d][t,u]

where **r** indicates a raw (character) interface to the diskette, **rdsk** selects the raw device interface and **dsk** selects the block device interface. **0** or **1** selects the drive to be accessed: **f0** selects floppy drive 0, while **f1** selects drive 1. The following list describes the format to be interacted with:

5h	5.25" high density diskette (1.2MB).
5d9	5.25" double density diskette, 9 sectors per track (360KB).
5d8	5.25" double density diskette, 8 sectors per track (320KB).

5d4	5.25" double density diskette, 4 sectors per track (320KB).
5d16	5.25" double density diskette, 16 sectors per track (320KB).
5q	5.25" quad density diskette (720KB).
3h	3.50" high density diskette (1.44MB).
3d	3.50" double density diskette (720KB).

Format specification is mandatory when opening the device for formatting. However, when accessing a floppy disk for other operations (read and write), the format specification field can be omitted. In this case, the floppy disk driver will automatically determine the format previously established on the diskette and then perform the requested operation (for example, **cpio -itv</dev/rdsk/f1**).

The last parameter, **t** or **u**, selects the partition to be accessed. **t** represents the whole diskette. Without **t** or **u** specified, the whole diskette except cylinder 0 will be selected. **u** represents the whole diskette except track 0 of cylinder 0.

Besides the device file naming convention described above, some of the formats have alias names that correlate to previous releases. The following list describes the formats that have an alias:

format	alias
5h	q15d
5d8	d8d
5d9	d9d

For example, the device file **/dev/rdsk/f0q15dt** is equivalent to **/dev/rdsk/f05ht**.

Files

XENIX Devices:

/dev/[r]fd0	/dev/[r]fd048ss8	/dev/[r]fd096	/dev/[r]fd0135ds9
/dev/[r]fd1	/dev/[r]fd148ss8	/dev/[r]fd196	/dev/[r]fd1135ds9
/dev/[r]fd048	/dev/[r]fd048ds9	/dev/[r]fd096ds9	/dev/[r]fd0135ds18
/dev/[r]fd148	/dev/[r]fd148ds9	/dev/[r]fd196ds9	/dev/[r]fd1135ds18
/dev/[r]fd048ds8	/dev/[r]fd048ss9	/dev/[r]fd096ds15	
/dev/[r]fd148ds8	/dev/[r]fd148ss9	/dev/[r]fd196ds15	

UNIX Devices:

/dev/[r]dsk/f0	/dev/[r]dsk/f05d9t	/dev/[r]dsk/f05d16	/dev/[r]dsk/f03ht
/dev/[r]dsk/f0t	/dev/[r]dsk/f0fd8	/dev/[r]dsk/f05d16t	/dev/[r]dsk/f03d
/dev/[r]dsk/f05h	/dev/[r]dsk/f05d8t	/dev/[r]dsk/f05q	/dev/[r]dsk/f03dt
/dev/[r]dsk/f05ht	/dev/[r]dsk/f05d4	/dev/[r]dsk/f05qt	
/dev/[r]dsk/f05d9	/dev/[r]dsk/f05d4t	/dev/[r]dsk/f03h	

Notes

It is not advisable to format a low density (48tpi) diskette on a high density (96tpi or 135tpi) floppy drive. Low density diskettes written on a high density drive should be read on high density drives. They may or may not be readable on a low density drive.

Use error-free floppy disks for best results on reading and writing.

hd

internal hard disk drive

Description

Device Filenames

There are two formats for the device filename describing a hard disk is, the first (**hd***dpv*) is based on XENIX, and is the preferred form. The second example is the equivalent UNIX form:

/dev/[r]hd*dpv*

/dev/[r]dsk/*dsp*

where:

[r] raw (character) special file or (or directory containing such files). If not present, this file is a block special file.

d disk "number" (actually a letter) ranging from **a** to **z**, and **A** to **Z**. The disk number is based on the order in which the disk was added to the system, not its physical location.

p *fdisk* partition number, from **1** to a maximum of **4**. Partition number **0** equals all partitions (the entire disk). The active partition is partition **a**. A DOS partition is partition **d**. (An active UNIX partiton is also partition 5; an active DOS parition, partition 6. The use of these partition numbers is discouraged. Use the equivalent letters described above.)

v is the *divvy* division number, numbered **0** to **6**. Division number **7** equals all divisions (the entire partition).

See *divvy*(ADM) for details on divisions, and *fdisk*(ADM) for details on partitions.

Block-buffered access to the first hard disk is provided through the following block special files: **hda0**, **hda1** through **hda4**, **hdaa** and **hdad**, **root**, and **swap**. Block-buffered access to the second hard disk is provided through the following block special files: **hdb0**, **hdb1** through **hdb4**, **hdba**.

hda0 refers to the entire physical disk; **hda1** through **hda4** refer to the fdisk partitions. **root** refers to the root file system; **swap** refers to the swap area; The block special files access the disks via the system's normal buffering mechanism and may be read and written without regard to the size of physical disk records.

Note

These disk names (and all other references to multiple hard disk configurations) are based on the the chronological sequence of disk additions, not necessarily the the physical order. Thus, when we indicate that **hda0** refers to the "first" hard disk, this means the disk was the first disk added to the system. The name **hdb0** refers to the "second" disk added to the system, which could possibly be located above the "first" disk. See the *System Administrator's Guide* for more information on hard disk naming conventions.

Character special files follow the same naming convention as the block special files except that the character special file is prefaced with an "r". For example, the character special file referring to the entire physical disk is **/dev/rhda0**.

The following are the names of the fixed disk partitions. Each partition can be accessed through a block interface, for example **/dev/hda1**, or through a character (raw) interface, for example **/dev/rhda1**.

The devices described above follow the XENIX naming convention. Equivalant UNIX devices are found in the **/dev/dsk** (character) and **/dev/rdsk** (raw) directories. In the table that follows, equivalent XENIX and UNIX device names are shown for various possible partitions of a hard disk.

Device Filenames for Hard Disks				
XENIX Naming		UNIX Naming		
Disk 1	Disk 2	Disk 1	Disk 2	Partition
/dev/hda0 /dev/rhda0	/dev/hdb0 /dev/rhdb0	/dev/dsk/as0 /dev/rdsk/as0	/dev/dsk/bs0 /dev/rdsk/bs0	entire disk
/dev/hda1 /dev/rhda1	/dev/hdb1 /dev/rhdb1	/dev/dsk/as1 /dev/rdsk/as1	/dev/dsk/bs1 /dev/rdsk/bs1	first partition
/dev/hda2 /dev/rhda2	/dev/hdb2 /dev/rhdb2	/dev/dsk/as2 /dev/rdsk/as2	/dev/dsk/bs2 /dev/rdsk/bs2	second partition
/dev/hda3 /dev/rhda3	/dev/hdb3 /dev/rhdb3	/dev/dsk/as3 /dev/rdsk/as3	/dev/dsk/bs3 /dev/rdsk/bs3	third partition
/dev/hda4 /dev/rhda4	/dev/hdb4 /dev/rhdb4	/dev/dsk/as4 /dev/rdsk/as4	/dev/dsk/bs4 /dev/rdsk/bs4	fourth partition
/dev/hdaa /dev/rhdaa	/dev/hdba /dev/rhdba	/dev/dsk/asa /dev/rdsk/asa	/dev/dsk/bsa /dev/rdsk/bsa	active partition
/dev/hdad /dev/rhdad	/dev/hdbd /dev/rhdbd	/dev/dsk/asd /dev/rdsk/asd	/dev/dsk/bsd /dev/rdsk/bsd	DOS partition
/dev/root /dev/rroot				root file system
/dev/swap /dev/rswap				swap area
/dev/usr /dev/rusr				user filesystem

Note that the **root**, **swap**, and **usr** device names exist only for the root disk. Also recall that the names of these disks (i.e., **hda0** or **as0**) are based on the assumption that ''Disk 1'' was the first disk installed in the system, and that ''Disk 2'' was installed second.

To access DOS partitions, specify letters such as ''C:'' or ''D:'' to indicate first or second partitions. The file **/etc/default/msdos** contains lines that assign a letter abbreviation for the DOS device name. Refer to *dos*(C).

Major/Minor Device Numbers

The operating system reserves 16 major device numbers, 64 through 79, in the kernel for use with hard disks. Since only 52 hard disks are currently supported, only major numbers 64 through 76 can be used. The remaining major numbers (77 through 79) are still reserved by the kernel, and are unavailable for use at this time.

Refer to the description of *divvy*(ADM) for a table that shows the relationship of disks (identified by their logical SCSI index) to major numbers and host adapter/board combinations.

The lower four bits of the major device number and bits 7 and 6 of the minor device number also indicate the logical SCSI index.

The following table lists the minor device number definitions for the hard disk special files, along with examples. Note that the block and character special devices share the same minor device definition. The minor device number definition is as follows: bits 7 and 6 denote physical drive, bits 5-3 denote *fdisk* partition and bits 2-0 denote *divvy* partition.

The special device filenames for the two disks in the following table are possible names used only as a likely example. Recall that the disk device filename is not dictated by major/minor number, or by physical location in the computer system. Disk device filenames follow the sequence in which they are added to the system.

Minor Device Bits				
Phys. 7 6	Partition 5 4 3	divvy 2 1 0	Device special filename	Description
0 0	0 0 0	0 0 0	/dev/hda0	whole PD 0
0 1	0 0 0	0 0 0	/dev/hdb0	whole PD 1
1 0	0 0 0	0 0 0	/dev/hdc0	whole PD 2
1 1	0 0 0	0 0 0	/dev/hdd0	whole PD 3
0 0	0 0 1	1 1 1	/dev/hda1	PD 0, whole Part. 1
0 0	0 1 0	1 1 1	/dev/hda2	PD 0, whole Part. 2
0 0	0 1 1	1 1 1	/dev/hda3	PD 0, whole Part. 3
0 0	1 0 0	1 1 1	/dev/hda4	PD 0, whole Part. 4
0 0	1 0 1	1 1 1	/dev/hdaa	PD 0, whole active Part.
0 0	1 1 0	1 1 1	/dev/hdad	PD 0, whole DOS Part.
0 0	1 0 1	0 0 0	/dev/root	PD 0, active Part., Div. 0
0 0	1 0 1	0 0 1	/dev/swap	PD 0, active Part., Div. 1
0 0	1 0 1	0 1 0	/dev/usr	PD 0, active Part., Div. 2
0 0	1 0 1	1 1 0	/dev/recover	PD 0, active Part., Div. 6
0 1	0 0 1	1 1 1	/dev/hdb1	PD 1, whole Part. 1
0 1	0 1 0	1 1 1	/dev/hdb2	PD 1, whole Part. 2
0 1	0 1 1	1 1 1	/dev/hdb3	PD 1, whole Part. 3
0 1	1 0 0	1 1 1	/dev/hdb4	PD 1, whole Part. 4
0 1	1 0 1	1 1 1	/dev/hdba	PD 1, whole active Part.
0 1	1 1 0	1 1 1	/dev/hdbd	PD 1, whole DOS Part.
0 1	0 0 1	0 0 0	/dev/hdb10	PD 1, Part. 1, Div. 0
0 1	0 0 1	0 0 1	/dev/hdb11	PD 1, Part. 1, Div. 1
0 1	0 0 1	0 1 0	/dev/hdb12	PD 1, Part. 1, Div. 2
1 1	0 0 1	1 1 1	/dev/hdc1	PD 2, whole Part. 1
1 1	0 1 0	1 1 1	/dev/hdc2	PD 2, whole Part. 2
1 1	0 1 1	1 1 1	/dev/hdc3	PD 2, whole Part. 3
1 1	1 0 0	1 1 1	/dev/hdc4	PD 2, whole Part. 4
1 1	1 0 1	1 1 1	/dev/hdca	PD 2, whole active Part.
1 1	1 1 0	1 1 1	/dev/hdcd	PD 2, whole DOS Part.
1 1	0 0 1	1 1 1	/dev/hdd1	PD 3, whole Part. 1
1 1	0 1 0	1 1 1	/dev/hdd2	PD 3, whole Part. 2
1 1	0 1 1	1 1 1	/dev/hdd3	PD 3, whole Part. 3
1 1	1 0 0	1 1 1	/dev/hdd4	PD 3, whole Part. 4
1 1	1 0 1	1 1 1	/dev/hdda	PD 3, whole active Part.
1 1	1 1 0	1 1 1	/dev/hddd	PD 3, whole DOS Part.
KEY	Part. = Partition Div. = Division		PD = physical drive (in order added, not location)	

See Also

fdisk(ADM), badtrk(ADM), divvy(ADM), dos(C), mkdev(ADM), the *System Administrator's Guide*

Diagnostics

The following messages are among those that may be printed on the console:

invalid fixed disk parameter table

and:

error on fixed disk (minor *n*), block = *nnnnn*,
cmd=*nnnnn*, status=*nnnn*,
Sector = *nnnnn*, Cylinder/head = *nnnnn*

Possible reasons for the first error include:

- The kernel is unable to get drive specifications, such as number of heads, cylinders, and sectors per track, from the disk controller ROM.
- Improper configuration.
- The disk is not turned on.
- The disk is not supported.

The second error specifies the following information:

- *block* : The UNIX block number within the device.
- *cmd* : The last command sent to the disk controller.
- *status* : The error status from the disk controller.
- *Sector* and *Cylinder/head* specify the location of a possible flaw. This information is used with *badtrk*(ADM). (Applicable only with non-SCSI hard disks.)

Notes

On the first disk, **hda0** denotes the entire disk and is used to access the master boot block which includes the fdisk partition table. For the second disk, **hdb0** denotes the entire disk and is used to access its fdisk partition table. Do not write to **hdb0** and **hda0**.

You can mount a filesystem only by referring to it by its full device filename (that must inlcude a divvy specification). For example, use **/dev/hdb10** to mount a filesystem that starts at division 0 of the active partiton (1) on the second hard disk (b). The device filename **/dev/hdb0**, to indicate all partitions (0) of the second disk (b), will not work!

keyboard

the PC keyboard

Description

The PC keyboard is used to enter data, switch screens, and send certain control signals to the computer. The Operating System performs terminal emulation on the PC screen and keyboard, and, in doing so, makes use of several particular keys and key combinations. These keys and key combinations have special names that are unique to UNIX systems, and may or may not correspond to the keytop labels on your keyboard. These keys are described later.

When you press a key, one of the following happens:

- An ASCII value is entered
- A string is sent to the computer.
- A function is initiated.
- The meaning of another key, or keys, is changed.

When a key is pressed (a keystroke), the keyboard sends a scancode to the computer, it is interpreted by the keyboard driver. The interpretation of key codes may be modified so that keys can function differently from their default actions.

There are three special occurrences, or keystrokes:

- Switch screens.
- Send signals.
- Change the value of previous character, characters or string.

Switching Screens (Multiscreen)

To get to the next consecutive screen, enter **Ctrl-PrtSc** using the **Ctrl** key, and the **PrtSc** key. Any active screen may be selected by entering **alt-F***n*, where **Fn** is one of the function keys. **F1** refers to the PC display (**/dev/tty01**).

Signals

A signal affects some process or processes. Examples of signals are **Ctrl-d** (end of input, exits from shell), **Ctrl-** (quits a process), **Ctrl-s** (stop output to the screen), and **Ctrl-q** (resume sending output).

Typically, characters are mapped to signals using *stty*(C). The only way to map signals is using *stty*.

Altering Values

The actual code sent to the keyboard driver can be changed by using certain keys in combination. For example, the SHIFT key changes the ASCII values of the alphanumeric keys. Holding down the **Ctrl** key while pressing another key sends a control code (**Ctrl-d**, **Ctrl-s**, **Ctrl-q**, etc.).

Special Keys

To help you find the special keys, the following table shows which keys on a typical console correspond to UNIX system keys. In this table, a hyphen (-) between keys means 'hold down the first key while pressing the second.'

UNIX Name	Keytop	Action
INTR	Del	Stops current action and returns to the shell. This key is also called the RUB OUT or INTERRUPT key.
BACKSPACE	←	Deletes the first character to the left of the cursor. Note that the "cursor left" key also has a left arrow (←) on its keytop, but you cannot backspace using that key.
Ctrl-d	Ctrl-d	Signals the end of input from the keyboard; also exits current shell.
Ctrl-h	Ctrl-h	Deletes the first character to the left of the cursor. Also called the ERASE key.
Ctrl-q	Ctrl-q	Restarts printing after it has been stopped with Ctrl-s.

UNIX Name	Keytop	Action
Ctrl-s	Ctrl-s	Suspends printing on the screen (does not stop the program).
Ctrl-u	Ctrl-u	Deletes all characters on the current line. Also called the KILL key.
Ctrl-\	Ctrl-\	Quits current command and creates a *core* file, if allowed. (Recommended for debugging only.)
ESCAPE	Esc	Special code for some programs. For example, changes from insert mode to command mode in the *vi*(C) text editor.
RETURN	(down-left arrow or ENTER)	Terminates a command line and initiates an action from the shell.
F*n*	F*n*	Function key *n*. F1-F12 are unshifted, F13-F24 are shifted F1-F12, F25-F36 are Ctrl-F1 through F12, and F37-F48 are Ctrl-Shift-F1 through F12. The next F*n* keys (F49-F60) are on the number pad (unshifted): F49 - '7' F55 - '6' F50 - '8' F56 - '+' F51 - '9' F57 - '1' F52 - '-' F58 - '2' F53 - '4' F59 - '3' F54 - '5' F60 - '0' For keys F61 through F96, see **/usr/lib/keyboard/strings.** These function keys are not available on all keyboards, but you can map other keys to represent them.

The keyboard mapping is performed through a structure defined in **/usr/include/sys/keyboard.h**. Each key can have ten states. The first eight are:

- Base	- Ctrl-Shift
- Shift	- Alt-Shift
- Ctrl	- Alt-Ctrl
- Alt	- Alt-Ctrl-Shift

There are two additional states indicated by two special bytes. The first is a "special state" byte whose bits indicate whether the key is "special" in one or more of the first eight states.

The second is one of four characters (C, N, B, O) which indicate how the lock keys affect the particular key. This is discussed further in the next section, "Scan Codes."

Keyboard Mode

Most keyboards normally are in a PC compatibility mode, though some can be put into a native AT keyboard mode. The UNIX utility *kbmode*(ADM) can be used to determine if a keyboard supports AT mode, and can also be used to put the keyboard into AT mode until the next time the system is rebooted. A system can also be configured to boot with the keyboard in AT mode with the *configure*(ADM) utility.

Enhanced keyboards are more fully programmable in AT mode. Also, it recognizes two control keys and an alt key.

Scan Codes

The following table describes the default contents of **/usr/lib/keyboard/keys**. The column headings are:

SCAN CODE - The scan code generated by the keyboard hardware when a key is pressed. There is no user access to the scan code generated by releasing a key.

BASE - The normal value of a key press.

SHIFT - The value of a key press when the SHIFT is also being held down.

LOCK - Indicates which lock keys affect that particular key:

- C indicates Capslock
- N indicates Numlock
- B indicates both
- O indicates locking is off

Keys affected by the lock keys C, B, or N, send the shifted value (scan code) of current state when that lock key is on. When the shift key is depressed while a lock key is also on, the key reverts (toggles) to its original state.

The other columns are the values of key presses when combinations of the CTRL, ALT and SHIFT keys are also held down.

All values, except for keywords, are ASCII character values. The keywords refer to the special function keys.

SCAN CODE	BASE	SHIFT	CTRL	CTRL SHIFT	ALT	ALT SHIFT	ALT CTRL	ALT CTRL SHIFT	LOCK
0	nop	nop	nop	nop	nop	nop	nop	nop	O
1	esc	esc	nop	nop	esc	esc	nop	nop	O
2	’1’	’!’	nop	nop	’1’	’!’	nop	nop	O
3	’2’	’@’	nop	nop	’2’	’@’	nop	nop	O
4	’3’	’#’	nop	nop	’3’	’#’	nop	nop	O
5	’4’	’$’	nop	nop	’4’	’$’	nop	nop	O
6	’5’	’%’	nop	nop	’5’	’%’	nop	nop	O
7	’6’	’^’	rs	rs	’6’	’^’	rs	rs	O
8	’7’	’&’	nop	nop	’7’	’&’	nop	nop	O
9	’8’	’*’	nop	nop	’8’	’*’	nop	nop	O
10	’9’	’(’	nop	nop	’9’	’(’	nop	nop	O
11	’0’	’)’	nop	nop	’0’	’)’	nop	nop	O
12	’-’	’_’	ns	ns	’-’	’_’	ns	ns	O
13	’=’	’+’	nop	nop	’=’	’+’	nop	nop	O
14	bs	bs	del	del	bs	bs	del	del	O
15	ht	btab	nop	nop	ht	btab	nop	nop	O
16	’q’	’Q’	dc1	dc1	’q’	’Q’	dc1	dc1	C
17	’w’	’W’	etb	etb	’w’	’W’	etb	etb	C
18	’e’	’E’	enq	enq	’e’	’E’	enq	enq	C
19	’r’	’R’	dc2	dc2	’r’	’R’	dc2	dc2	C
20	’t’	’T’	dc4	dc4	’t’	’T’	dc4	dc4	C
21	’y’	’Y’	em	em	’y’	’Y’	em	em	C
22	’u’	’U’	nak	nak	’u’	’U’	nak	nak	C
23	’i’	’I’	ht	ht	’i’	’I’	ht	ht	C
24	’o’	’O’	si	si	’o’	’O’	si	si	C
25	’p’	’P’	dle	dle	’p’	’P’	dle	dle	C
26	’[’	’{’	esc	esc	’[’	’{’	esc	esc	O
27	’]’	’}’	gs	gs	’]’	’}’	gs	gs	O
28	cr	cr	nl	nl	cr	cr	nl	nl	O
29	ctrl	ctrl	ctrl	ctrl	ctrl	ctrl	ctrl	ctrl	O
30	’a’	’A’	soh	soh	’a’	’A’	soh	soh	C
31	’s’	’S’	dc3	dc3	’s’	’S’	dc3	dc3	C
32	’d’	’D’	eot	eot	’d’	’D’	eot	eot	C
33	’f’	’F’	ack	ack	’f’	’F’	ack	ack	C
34	’g’	’G’	bel	bel	’g’	’G’	bel	bel	C
35	’h’	’H’	bs	bs	’h’	’H’	bs	bs	C
36	’j’	’J’	nl	nl	’j’	’J’	nl	nl	C
37	’k’	’K’	vt	vt	’k’	’K’	vt	vt	C
38	’l’	’L’	np	np	’l’	’L’	np	np	C
39	’;’	’:’	nop	nop	’;’	’:’	nop	nop	O
40	’\’’	’"’	nop	nop	’\’’	’"’	nop	nop	O
41	’`’	’~’	nop	nop	’`’	’~’	nop	nop	O

SCAN CODE	BASE	SHIFT	CTRL	CTRL SHIFT	ALT	ALT SHIFT	ALT CTRL	ALT CTRL SHIFT	LOCK
42	lshift	lshift	lshift	lshift	lshift	lshift	lshift	lshift	O
43	’\\’	’|’	fs	fs	’\\’	’|’	fs	fs	O
44	’z’	’Z’	sub	sub	’z’	’Z’	sub	sub	C
45	’x’	’X’	can	can	’x’	’X’	can	can	C
46	’c’	’C’	etx	etx	’c’	’C’	etx	etx	C
47	’v’	’V’	syn	syn	’v’	’V’	syn	syn	C
48	’b’	’B’	stx	stx	’b’	’B’	stx	stx	C
49	’n’	’N’	so	so	’n’	’N’	so	so	C
50	’m’	’M’	cr	cr	’m’	’M’	cr	cr	C
51	’,’	’<’	nop	nop	’,’	’<’	nop	nop	O
52	’.’	’>’	nop	nop	’.’	’>’	nop	nop	O
53	’/’	’?’	nop	nop	’/’	’?’	nop	nop	O
54	rshift	rshift	rshift	rshift	rshift	rshift	rshift	rshift	O
55	’*’	’*’	nscr	nscr	’*’	’*’	nscr	nscr	O
56	alt	alt	alt	alt	alt	alt	alt	alt	O
57	’ ’	’ ’	’ ’	’ ’	’ ’	’ ’	’ ’	’ ’	O
58	clock	clock	clock	clock	clock	clock	clock	clock	O
59	fkey1	fkey13	fkey25	fkey37	scr1	scr11	scr1	scr11	O
60	fkey2	fkey14	fkey26	fkey38	scr2	scr12	scr2	scr12	O
61	fkey3	fkey15	fkey27	fkey39	scr3	scr13	scr3	scr13	O
62	fkey4	fkey16	fkey28	fkey40	scr4	scr14	scr4	scr14	O
63	fkey5	fkey17	fkey29	fkey41	scr5	scr15	scr5	scr15	O
64	fkey6	fkey18	fkey30	fkey42	scr6	scr16	scr6	scr16	O
65	fkey7	fkey19	fkey31	fkey43	scr7	scr7	scr7	scr7	O
66	fkey8	fkey20	fkey32	fkey44	scr8	scr8	scr8	scr8	O
67	fkey9	fkey21	fkey33	fkey45	scr9	scr9	scr9	scr9	O
68	fkey10	fkey22	fkey34	fkey46	scr10	scr10	scr10	scr10	O
69	nlock	nlock	dc3	dc3	nlock	nlock	dc3	dc3	O
70	slock	slock	del	del	slock	slock	del	del	O
71	fkey49	’7’	’7’	’7’	’7’	’7’	’7’	’7’	N
72	fkey50	’8’	’8’	’8’	’8’	’8’	’8’	’8’	N
73	fkey51	’9’	’9’	’9’	’9’	’9’	’9’	’9’	N
74	fkey52	’-’	’-’	’-’	’-’	’-’	’-’	’-’	N
75	fkey53	’4’	’4’	’4’	’4’	’4’	’4’	’4’	N
76	fkey54	’5’	’5’	’5’	’5’	’5’	’5’	’5’	N
77	fkey55	’6’	’6’	’6’	’6’	’6’	’6’	’6’	N
78	fkey56	’+’	’+’	’+’	’+’	’+’	’+’	’+’	N
79	fkey57	’1’	’1’	’1’	’1’	’1’	’1’	’1’	N
80	fkey58	’2’	’2’	’2’	’2’	’2’	’2’	’2’	N
81	fkey59	’3’	’3’	’3’	’3’	’3’	’3’	’3’	N
82	fkey60	’0’	’0’	’0’	’0’	’0’	’0’	’0’	N
83	del	’.’	del	del	del	del	del	del	N
84	nop	nop	nop	nop	nop	nop	nop	nop	O
85	fkey11	fkey23	fkey35	fkey47	scr11	scr11	scr11	scr11	O
86	fkey12	fkey24	fkey36	fkey48	scr12	scr12	scr12	scr12	O

The following scan codes exist only for keyboards which support, and are in, native AT mode rather than PC compatibility mode.

SCAN CODE	BASE	SHIFT	CTRL	CTRL SHIFT	ALT	ALT SHIFT	ALT CTRL	ALT CTRL SHIFT	LOCK
87	fkey11	fkey23	fkey35	fkey47	scr11	scr11	scr11	scr11	O
88	fkey12	fkey24	fkey36	fkey48	scr12	scr12	scr12	scr12	O
89	nop	nop	nop	nop	nop	nop	nop	nop	O
90	nop	nop	nop	nop	nop	nop	nop	nop	O
91	nop	nop	nop	nop	nop	nop	nop	nop	O
92	nop	nop	nop	nop	nop	nop	nop	nop	O
93	nop	nop	nop	nop	nop	nop	nop	nop	O
94	nop	nop	nop	nop	nop	nop	nop	nop	O
95	nop	nop	nop	nop	nop	nop	nop	nop	O
96	fkey50	fkey50	fkey50	fkey50	fkey50	fkey50	fkey50	fkey50	O
97	fkey53	fkey53	fkey53	fkey53	fkey53	fkey53	fkey53	fkey53	O
98	fkey58	fkey58	fkey58	fkey58	fkey58	fkey58	fkey58	fkey58	O
99	fkey55	fkey55	fkey55	fkey55	fkey55	fkey55	fkey55	fkey55	O
100	fkey49	fkey49	fkey49	fkey49	fkey49	fkey49	fkey49	fkey49	O
101	fkey51	fkey51	fkey51	fkey51	fkey51	fkey51	fkey51	fkey51	O
102	fkey57	fkey57	fkey57	fkey57	fkey57	fkey57	fkey57	fkey57	O
103	fkey59	fkey59	fkey59	fkey59	fkey59	fkey59	fkey59	fkey59	O
104	fkey60	fkey60	fkey60	fkey60	fkey60	fkey60	fkey60	fkey60	O
105	del	del	del	del	del	del	del	del	N
106	fkey54	fkey54	fkey54	fkey54	fkey54	fkey54	fkey54	fkey54	O
107	nop	nop	nop	nop	nop	nop	nop	nop	O
108	nop	nop	nop	nop	nop	nop	nop	nop	O
109	nop	nop	nop	nop	nop	nop	nop	nop	O
110	nop	nop	nop	nop	nop	nop	nop	nop	O
111	nop	nop	nop	nop	nop	nop	nop	nop	O
112	nop	nop	nop	nop	nop	nop	nop	nop	O
113	nop	nop	nop	nop	nop	nop	nop	nop	O
114	nop	nop	nop	nop	nop	nop	nop	nop	O
115	nop	nop	nop	nop	nop	nop	nop	nop	O
116	nop	nop	nop	nop	nop	nop	nop	nop	O
117	nop	nop	nop	nop	nop	nop	nop	nop	O
118	nop	nop	nop	nop	nop	nop	nop	nop	O
119	nop	nop	nop	nop	nop	nop	nop	nop	O
120	nop	nop	nop	nop	nop	nop	nop	nop	O
121	nop	nop	nop	nop	nop	nop	nop	nop	O
122	nop	nop	nop	nop	nop	nop	nop	nop	O
123	nop	nop	nop	nop	nop	nop	nop	nop	O
124	nop	nop	nop	nop	nop	nop	nop	nop	O
125	nop	nop	nop	nop	nop	nop	nop	nop	O

SCAN CODE	BASE	SHIFT	CTRL	CTRL SHIFT	ALT	ALT SHIFT	ALT CTRL	ALT CTRL SHIFT	LOCK
126	nop	nop	nop	nop	nop	nop	nop	nop	O
127	nop	nop	nop	nop	nop	nop	nop	nop	O
128	rctrl	rctrl	rctrl	rctrl	rctrl	rctrl	rctrl	rctrl	O
129	ralt	ralt	ralt	ralt	ralt	ralt	ralt	ralt	O
130	fkey60	fkey60	fkey60	fkey60	fkey60	fkey60	fkey60	fkey60	O
131	del	del	del	del	del	del	del	del	N
132	fkey49	fkey49	fkey49	fkey49	fkey49	fkey49	fkey49	fkey49	O
133	fkey57	fkey57	fkey57	fkey57	fkey57	fkey57	fkey57	fkey57	O
134	fkey51	fkey51	fkey51	fkey51	fkey51	fkey51	fkey51	fkey51	O
135	fkey59	fkey59	fkey59	fkey59	fkey59	fkey59	fkey59	fkey59	O
136	fkey53	fkey53	fkey53	fkey53	fkey53	fkey53	fkey53	fkey53	O
137	fkey55	fkey55	fkey55	fkey55	fkey55	fkey55	fkey55	fkey55	O
138	fkey50	fkey50	fkey50	fkey50	fkey50	fkey50	fkey50	fkey50	O
139	fkey58	fkey58	fkey58	fkey58	fkey58	fkey58	fkey58	fkey58	O
140	'/'	nop	nop	nop	'/'	nop	nop	nop	O
141	cr	cr	nl	nl	cr	cr	nl	nl	O

The next table lists the "value" of each of the special keywords used in **/usr/lib/keyboard/keys** (and the preceding table). *mapkey*(M) places a "value" in the *ioctl* buffer during key mapping. The keywords are only used in the scan code file (**/usr/lib/keyboard/keys**) for readability.

Name	Value	Meaning
nop	0	No operation - no action from keypress
lshift	2	Left hand shift
rshift	3	Right hand shift
clock	4	Caps lock
nlock	5	Numeric lock
slock	6	Scroll lock
alt	7	Alt key
btab	8	Back tab key - generates fixed sequence (esc [Z)
ctrl	9	Control key
nscr	10	Switch to the next screen
scr1	11	Switch to screen #1
...		...
scr16	26	Switch to screen #16
fkey1	27	Function key #1
...		...
fkey96	122	Function key #96
rctl	128*	Right Control Key
ralt	129*	Right Alt Key

* AT mode keyboard only.

This table lists names and decimal values that are interchangeable in the *mapkey* file. Names are used in place of numeric constants to make it easier to read the scan code table. Again, only the decimal values are placed in the *ioctl* buffer. These are taken from **ascii**(M).

Name	Value	Name	Value
nul	0	dc1	17
soh	1	dc2	18
stx	2	dc3	19
etx	3	dc4	20
eot	4	nak	21
enq	5	syn	22
ack	6	etb	23
bel	7	can	24
bs	8	em	25
ht	9	sub	26
nl	10	esc	27
vt	11	fs	28
np	12	gs	29
cr	13	rs	30
so	14	ns	31
si	15	del	127
dle	16		

Keyboard Mapping

The PC keyboard is mapped as part of terminal emulation. This kind of mapping is performed only on the computer keyboard, not on remote terminals. Use *mapkey* to change keyboard mapping. To change the mapping for individual channels (multiscreens), use *mapchan*(M).

Keyboard mapping can also be performed using *ioctl*. The syntax is the same as for string key mapping (see previous section).

For keyboard mapping, *cmd* is GIO_KEYMAP to display the current map, and PIO_KEYMAP puts the prepared buffer into place.

String Key Mapping

To map string (function) keys, use the *mapstr* (see *mapkey*(M)) utility. *mapstr* modifies the string mapping table where function keys are defined.

The string mapping table is an array of 512 bytes (typedef *strmap_t*) containing null terminated strings that redefine the function keys. The first null terminated string is assigned to the first string key, the second string to the second string key, and so on.

There is no limit to the length of any particular string as long as the whole table does not exceed 512 bytes, including nulls. Strings are made null by the introduction of extra null characters.

The following is a list of default function key values:

Default Function Key Values		
Key Number	Function Key	Function
1	F1	ESC [M
2	F2	ESC [N
3	F3	ESC [O
4	F4	ESC [P
5	F5	ESC [Q
6	F6	ESC [R
7	F7	ESC [S
8	F8	ESC [T
9	F9	ESC [U
10	F10	ESC [V
11	F11	ESC [W
12	F12	ESC [X
13	Shift-F1	ESC [Y
14	Shift-F2	ESC [Z
15	Shift-F3	ESC [a
16	Shift-F4	ESC [b
17	Shift-F5	ESC [c
18	Shift-F6	ESC [d
19	Shift-F7	ESC [e
20	Shift-F8	ESC [f
21	Shift-F9	ESC [g
22	Shift-F10	ESC [h
23	Shift-F11	ESC [i
24	Shift-F12	ESC [j
25	Ctrl-F1	ESC [k
26	Ctrl-F2	ESC [l
27	Ctrl-F3	ESC [m
28	Ctrl-F4	ESC [n
29	Ctrl-F5	ESC [o
30	Ctrl-F6	ESC [p
31	Ctrl-F7	ESC [q
32	Ctrl-F8	ESC [r
33	Ctrl-F9	ESC [s
34	Ctrl-F10	ESC [t
35	Ctrl-F11	ESC [u
36	Ctrl-F12	ESC [v

Default Function Key Values (continued)		
Key Number	**Function Key**	**Function**
37	Ctrl-Shift-F1	ESC [w
38	Ctrl-Shift-F2	ESC [x
39	Ctrl-Shift-F3	ESC [y
40	Ctrl-Shift-F4	ESC [z
41	Ctrl-Shift-F5	ESC [@
42	Ctrl-Shift-F6	ESC [[
43	Ctrl-Shift-F7	ESC [\
44	Ctrl-Shift-F8	ESC []
45	Ctrl-Shift-F9	ESC [^
46	Ctrl-Shift-F10	ESC [_
47	Ctrl-Shift-F11	ESC [`
48	Ctrl-Shift-F12	ESC [{
49	Home	ESC [H
50	Up arrow	ESC [A
51	Page up	ESC [I
52	Minus sign	-
53	Left arrow	ESC [D
54	5	ESC [E
55	Right arrow	ESC [C
56	Plus sign	+
57	End	ESC [F
58	Down arrow	ESC [B
59	Page down	ESC [G
60	Insert	ESC [L

You can also map string keys using *ioctl*(S). The syntax is:

```
#include <sys/keyboard.h>
ioctl(fd,cmd,buf)
int fd, cmd;
char *buf;
...
```

For string key mapping where *cmd* is GIO_STRMAP to display the string mapping table and PIO_STRMAP to put the new string mapping table in place.

Files

/usr/lib/keyboard/keys
/usr/lib/keyboard/strings

See Also

mapchan(F), mapchan(M), mapkey(M), multiscreen(M), screen(HW), setkey(C), stty(C), kbmode(ADM), configure(ADM)

lp, lp0

line printer device interfaces

Description

The **lp0** and **lp1** files provide access to the parallel ports of the computer.

Files

/dev/lp0

See Also

lp(C), lpadmin(ADM), lpsched(ADM), pcu(ADM)

Notes

The standard **lp** port, **lp0**, sends a printer initialization string the first time the file is opened after the system is *booted*.

mouse

system mouse

Description

Altos UNIX System V supports mice attached directly to controller cards on the bus and mice attached to standard serial ports. The command:

mkdev mouse

is used to configure a new mouse or to reconfigure an existing mouse.

See Also

mkdev(ADM), usemouse(C)

Files

/dev/mouse	Directory for mouse-related special device files.
/dev/mouse/bus[0-1]	Bus mouse device files.
/dev/mouse/vpix[0-1]	vpix-mouse device files.
/dev/mouse/microsoft_ser	Microsoft serial mouse device files.
/dev/mouse/logitech_ser	Logitech serial mouse device files.
/dev/mouse/mousesys_ser	Mousesys serial mouse device files.
/dev/mouse/ttyp[0-7]	Special pseudo-tty files for mouse input.
/etc/default/usemouse	Default map file for mouse-generated characters.
/usr/lib/event/devices	File containing device information for mice.
/usr/lib/event/ttys	File listing ttys eligible to use mice.
/usr/lib/mouse/*	Alternate map files for mice.

Value Added

mouse is an extension of AT&T System V provided in Altos UNIX System V.

parallel

parallel interface devices

Description

This section describes the parallel port (LPT1) on the Base I/O Board:

/dev/lp0 Parallel port device

For information on the TCU/2 parallel ports, refer to the appropriate TCU/2 documentation.

If a parallel device fails to interrupt properly, the parallel driver enters "poll mode." Once interrupts are received from the device, the driver returns to its original mode.

The parallel driver delays a certain amount of time when a parallel device is closed. The amount of delay can affect printer performance, but is necessary to compensate for different sizes of printer buffers and printer speeds. For example, this command sets the delay on close to 1 second, specified in 10ths of a second:

```
stty time 10< /dev/lp0
```

When given from a prompt, this command will only work if the port is open. It is recommended that a variation of this command be placed in the interface script used with the parallel device to achieve the same results:

```
stty time 10 0< &1
```

Notes

Parallel adapters on add-on cards will function, but switches must be set correctly.

In most cases, port control is best handled with the port configuration utility, *pcu*(ADM).

The *stty*(C) command for output processing is supported on a parallel device. *stty* options that have no effect on a parallel device are ignored and no error messages are displayed.

Usage

Usually invoked by *lp*(C), but can be written to directly.

Files

/dev/lp0

See Also

lp(C), lp(HW), lpadmin(ADM), lpsched(ADM), serial(HW), pcu(ADM)

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 the system clock and samples the program counter at line frequency. 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(ADM)

ramdisk

memory block device

Description

The *ramdisk* device driver provides a block interface to memory. A *ramdisk* can be used like any other block device, including making it into a file systems using *mkfs*(ADM). There are eight *ramdisks* available.

The characteristics of a *ramdisk* file are determined by its minor device number. The bits in the minor device number encode its size, longevity, and which of the eight possible *ramdisks* it is.

The three low-order bits of the minor device number determine which of the eight *ramdisks* is being accessed.

The next four bits of the minor device number determine the size of the *ramdisk*. The size of a *ramdisk* must be a power of 2, and must be at least 16K. Since 4 bits are available, there are 16 possible sizes, starting at 16K and doubling every time the size indicator is incremented, to produce possible sizes of 16K, 32K, 64K, and up.

The high-order bit is a longevity indicator. If set, memory is permanently allocated to that *ramdisk*, and can be deallocated only by rebooting the system. Permanent *ramdisks* can only be allocated by the superuser. However, once a permanent *ramdisk* is allocated (by opening it), it can be read and written by anyone having the appropriate permissions on the *ramdisk* inode.

If clear, the *ramdisk* is deallocated when no processes have it open. To create an easily removable, but semi-permanent *ramdisk*, use a separate process to keep the device open for as long as necessary.

Since a complete set of *ramdisks* (8) would consume 256 inodes, only one example 16K *ramdisk* (**/dev/ram00**) is created when the system is installed. The system administrator can check this existing file to determine the major device number for any other required *ramdisks*. All *ramdisks* will use the same major device number.

The following table shows how the minor device number is constructed:

Example Minor Device Number Construction									
Description	Longe-vity	Size (see next table)				Ram Disk No.			Minor Device Number
16K (#1) (Temporary)	0	0	0	0	0	0	0	1	1
16K (#1) (Permanent)	1	0	0	0	0	0	0	1	129
64K (#0) (Temporary)	0	0	0	1	0	0	0	0	16
512K (#7) (Permanent)	1	0	1	0	1	1	1	1	175

The contents of the size field and the corresponding *ramdisk* size is shown in the next table.

Size Bits				Ramdisk Size
0	0	0	0	16K
0	0	0	1	32K
0	0	1	0	64K
0	0	1	1	128K
0	1	0	0	256K
0	1	0	1	512K
0	1	1	0	1M
0	1	1	1	2M
1	0	0	0	4M
1	0	0	1	8M
1	0	1	0	16M
1	0	1	1	32M
1	1	0	0	64M
1	1	0	1	128M
1	1	1	0	256M
1	1	1	1	512M

To create a ramdisk, follow these steps:

1. Create the device node.

You must first create the device that the ramdisk will reside on. It has the form:

mknod *device_name b_or_c major_device_number minor_device_number*

where *b_or_c* ''b'' or ''c''. ''b'' is for blocked devices and is the one you will use. The major number will always be 31. The minor number is derived from the table above. The minor number is the sum of the three attribute columns.

Longevity:
permanent = 128 non-permanent = 0

Size:

16K = 0	128K = 24	1 Meg = 48	8 Meg = 72
32K = 8	256K = 32	2 Meg = 56	16 Meg = 80
64K = 16	512K = 40	4 Meg = 64	32 Meg = 88

Ram Disk number: 0 through 7 Note: There are only 8 devices available. Two different size devices may not share the same number.

For example, to create a 64K permanent ramdisk, the minor number could vary from 144 to 151. If the disk number was 1 the mknod command would be:

mknod /dev/ram64 b 31 145

2. Make a file system.

This creates a file system on the the ramdisk. In this example *mkfs* has the form:

mkfs *device_name size_of_file_in_Bsize_blocks*

In this example, the command to create a 64K file system would be:

mkfs /dev/ram64 64

3. Mount the filesystem.

This mounts the selected device on the specified mount point. It has the form:

mount device_name mount_point

In order to mount the example 64K ramdisk on /mnt the command would be:

mount /dev/ram64 /mnt

To make a file system on a non-permanent *ramdisk*, the device file must be held open between the *mkfs* and the *mount*(ADM) operations. Otherwise, the *ramdisk* is allocated at the start of the *mkfs* command, and deallocated at its end. Once the *ramdisk* is mounted, it is open until it is unmounted.

The following shell fragment shows one way to use *mkfs* on a non-permanent 512K *ramdisk*, then mount it:

```
(       /etc/mkfs /dev/ram40 512
        /etc/mount /dev/ram40 /mnt
) < /dev/ram40
```

Notes

ramdisks must occupy contiguous memory. If free memory is fragmented, opening a *ramdisk* may fail even though there is enough total memory available. Ideally, all *ramdisks* should be allocated at system startup. This helps prevent the *ramdisks* themselves from fragmenting memory.

ramdisks are geared towards use in specialized applications. In many cases, you will notice a *decrease in system performance* when *ramdisks* are used, because UNIX can generally put the memory to better use elsewhere.

Files

/dev/ram00

See Also

mkfs(ADM), mount(ADM), mknod(C)

Value Added

ramdisk is an extension of AT&T System V provided in Altos UNIX System V.

rtc

real time clock interface

Description

The **rtc** driver supports the real time clock chip, allowing it to be set with the correct local time and allowing the time to be read from the chip.

Ioctl Calls

RTCRTIME
This call is used to read the local time from the real time clock chip. The argument to the *ioctl* is the address of a buffer of **RTCNREG** unsigned characters (**RTCNREG** is defined as <sys/**rtc.h**>). The *ioctl* will fill in the buffer with the contents of the chip registers. Currently, **RTCNREG** is 14, and the meanings of the byte registers are as follows:

Register	Contents
0	Seconds
1	Second alarm
2	Minutes
3	Minute alarm
4	Hours
5	Hour alarm
6	Day of week
7	Date of month
8	Month
9	Year
A	Status register A
B	Status register B
C	Status register C
D	Status register D

For further information on the functions of these registers, see your hardware technical reference manual.

RTCSTIME
This call is used to set the time into the real time clock chip. The argument to the *ioctl* is the address of a buffer of **RTCNREGP** unsigned characters (**RTCNREGP** as defined in <sys/**rtc.h**>). These bytes should be the desired chip register contents. Currently, **RTCNREGP** is 10, representing registers 0-9 as shown above. Note that only the super-user may open the real time clock device for writing and that the **RTCSTIME** *ioctl* will fail for any other than the super-user.

Files

/dev/rtc

screen

tty[01-n], color, monochrome, ega, vga display adapter and video monitor

Description

The **tty[01-*n*]** device files provide character I/O between the system and the video display monitor and keyboard. Each file corresponds to a separate teletype device. Although there is a maximum of 12 screens, the exact number available (*n*) depends upon the amount of memory in the computer. The screens are modeled after a 25 line, 80 column ASCII terminal, unless specified otherwise.

System error messages from the kernel are written to **/dev/console**, which is normally the current multiscreen. If the **/dev/console** is the default output device for system error messages, and the display being used is switched to graphics mode, console messages are not displayed. When the video device returns to text mode, a notice message is displayed and the text of the kernel error can be recovered from **usr/adm/messages.**

Although all **tty**[01-*n*] devices may be open concurrently, only one of the corresponding devices can be active at any given time. The active device displays its own screen and takes sole possession of the keyboard. It is an error to attempt to access the **color**, **monochrome**, or **ega** file when no corresponding adapter exists or no multiscreens are associated with it.

To get to the next consecutive screen, enter **Ctrl-PrtSc** using the **Ctrl** key, and the **PrtSc** key. Any active screen may be selected by entering **alt-Fn**, where **Fn** is one of the function keys. For example, **F1** refers to the **tty01** device.

Control Modes

Multiscreens can be reassigned to different adapters (in multi-adapter systems) with these *ioctl*s :

SWAPMONO	Selects the monochrome display as the output device for the multiscreen.
SWAPCGA	Selects the regular color display as the output device for the multiscreen.

SWAPEGA	Selects the enhanced color display as the output device for the multiscreen.
SWAPVGA	Selects the video graphics array color display as the output device for the multiscreen.

To find out which display adapter type is currently attached to the multiscreen, you can use *ioctl*(S) with the following request:

CONS_CURRENT	Returns the display adapter type currently associated with the multiscreen. The return value can be one of: MONO, CGA, EGA, or VGA.

Display Modes

The following *ioctl*s can be used to change the video display mode:

SW_B80x25	Selects 80x25 black and white text display mode. (MONO, CGA, EGA, VGA)
SW_C80x25	Selects 80x25 color text display mode. (CGA, EGA, VGA)
SW_B40x25	Selects 40x25 black and white text display mode. (MONO, CGA, EGA, VGA)
SW_C40x25	Selects 40x25 color text display mode. (CGA, EGA, VGA)
SW_BG320	Selects 320x200 black and white graphics display mode. (CGA, EGA, VGA)
SW_CG320	Selects 320x200 color graphics display mode. (CGA, EGA, VGA)
SW_BG640	Selects 640x200 black and white graphics display mode. (CGA, EGA, VGA)
SW_EGAMONO80x25	Selects EGA (Enhanced Graphics Adapter) mode 7 - emulates support provided by the monochrome display. (EGA, VGA)

SW_EGAMONOAPA	Selects EGA support for 640x350 graphics display mode (EGA mode F). (EGA with mono monitor)
SW_ENHMONOAPA2	Selects EGA mode F*. (EGA with mono monitor)
SW_ENHB40x25	Selects enhanced EGA support for 40x25 black and white text display mode. (EGA, VGA)
SW_ENHC40x25	Selects enhanced EGA support for the 40x25 color text display mode. (EGA, VGA)
SW_ENHB80x25	Selects enhanced EGA support for 80x25 black and white text display mode. (EGA, VGA)
SW_ENHC80x25	Selects enhanced EGA support for 80x25 color text display mode. (EGA, VGA)
SW_ENHB80x43	Selects enhanced EGA support for 80x43 black and white text display mode. (EGA, VGA)
SW_ENHC80x43	Selects enhanced EGA support for 80x43 color text display mode. (EGA, VGA)
SW_CG320_D	Selects EGA support for 320x200 graphics display mode. (EGA mode D.) (EGA, VGA)
SW_CG640_E	Selects EGA support for 640x200 graphics display mode (EGA mode E). (EGA, VGA)
SW_CG640x350	Selects EGA support for 640x350 graphics display mode (EGA mode 10). (EGA, VGA)
SW_ENH_CG640	Selects EGA mode 10*. (EGA, VGA)
SW_MCAMODE	Reinitializes the monochrome adapter. (MONO)
SW_VGA40x25	Selects VGA support for the 40x25 color text display mode (VGA mode 1+). (VGA)
SW_VGA80x25	Selects VGA support for the 80x25 black and white text display mode (VGA mode 2+). (VGA)

SW_VGAM80x25	Selects VGA mode 7+ - emulates support provided by the monochrome display. (VGA with mono monitor)
SW_VGA11	Selects VGA support for the 640x480 graphics display mode (VGA mode 11). (VGA)
SW_VGA12	Selects VGA support for the 640x480 graphics display mode (VGA mode 12). (VGA)
SW_VGA13	Selects VGA support for the 320x200 graphics display mode (VGA mode 13). (VGA)

Switching to an invalid display mode for a display device will result in an error.

Getting Display Modes

The following *ioctl*( ) requests are provided to obtain information about the current display modes:

CONS_GET	Returns the current display mode setting for current display adapter. (All)
CGA_GET	Returns the current display mode setting of the color graphics adapter. (CGA only)
EGA_GET	Returns the current display mode setting of the enhanced graphics adapter. (EGA only)
MCA_GET	Returns the current display mode setting of the monochrome adapter. (MONO only)
VGA_GET	Returns the current display mode of the video graphics adapters. (VGA only)
CONS_GETINFO	Returns structure *vid_info* (below). Size of structure (first field) must be filled in by user.

```
struct vid_info
{
      short    size;                 /* must be first field          */
      short    m_num;                /* multiscreen number, 0 based  */
      ushort   mv_row, mv_col;       /* cursor position              */
      ushort   mv_rsz, mv_csz;       /* text screen size             */
      struct   colors mv_norm,       /* normal attributes            */
                      mv_rev,        /* reverse video attributes     */
                      mv_grfc;       /* graphic character attributes */
      uchar_t mv_ovscan;             /* border color                 */
      uchar_t mk_keylock;            /* caps/num/scroll lock         */
};
```

CONS_6845INFO — Returns structure *m6845_info* (below). Size of structure (first field) must be filled in by user.

```
struct m6845_info
{
    short   size;           /* must be first field        */
    ushort  screen_top;     /* offset of screen in video  */
    ushort  cursor_type;    /* cursor shape               */
};
```

CONSADP — Returns number of multiscreen displayed on adaptor associated with that multiscreen.

GIO_ATTR — Return value of ioctl is 6845-style attribute byte in effect.

GIO_COLOR — Return value of ioctl is zero or one depending on whether the device supports color

GIO_SCRNMAP — Gets the 256-byte screen map table, which is the mapping of ASCII values (0-256) onto the PC video ROM font characters (0-256). Note that control characters (ASCII values less than hex 20) have control functions and do not display ROM characters (example: ^J is newline).

This is often used to map the low font values that normally correspond to ASCII control values to higher ASCII values, thus displaying the desired ROM characters.

PIO_SCRNMAP — Puts the 256-byte screen map table (see GIO_SCRNMAP).

PIO_KEYMAP — See *keyboard*(HW)

PIO_KEYMAP — See *keyboard*(HW)

GIO_FONT8X*n* — Gets font, where *n* is 8, 14, and 16. Argument is a pointer to a font table. Size of 8X8 font table is 8X256 bytes, 8X14 is 14X256 bytes, etc.

PIO_FONT8X*n* — Puts font, where *n* is 8, 14, and 16. Argument is a pointer to a font table. Size of 8X8 font table is 8X256 bytes, 8X14 is 14X256 bytes, etc.

Memory Mapping Modes

The *ioctl*(S) routine is used to map the display memory of the various devices into the user's data space.

Note that the MAP* ioctls map the memory associated with the current mode. You must put the adapter into the desired mode before performing mapping, or the pointers returned will not be appropriate. Refer to your hardware manual for details on various displays, adapters, and controllers.

These *ioctl*( ) requests can be used to map the display memory:

MAPCONS	Maps the display memory of the adaptor currently being used into the user's data space. (All)
MAPMONO	Maps the monochrome adapter's display memory into the user's data space. (MONO only)
MAPCGA	Maps the color adapter's display memory into the user's data space. (CGA only)
MAPEGA	Maps the enhanced graphics adapter's display memory into the user's data space. (EGA only)
MAPVGA	Maps the video graphics adapter's display memory into the user's data space. (VGA only)

For example, the following code can be used to acquire a pointer to the start of the user data space associated with the color graphics adapter display memory:

```
char *dp;
int retval;
      .
      .
      .
/* fd is a file descriptor for a
   multiscreen device */
retval = ioctl (fd, MAPCONS,0L);
dp = (char *) retval;
      .
      .
      .
```

Note that when the display memory is mapped into the user space, the adapter's m6845 start address registers are not set. The start address can be reset in two ways, so that the start address of the display memory corresponds to the upper left hand corner of the screen:

1. Switch modes with an *ioctl*() (the "switch" can be to the present mode). See the "Display Modes" section of this manual page.

2. Change the start address high and low address with the *in-on-port*/*out-on-port* *ioctl*().

The *in-on-port*/*out-on-port* ioctl()'s can also be used to determine the current value in the start address register, and then set up a pointer to point to the offset in the mapped-in data space.

MAP_CLASS
Package ioctl that gives I/O privileges to an arbitrary list of ports and maps an arbitrary frame buffer into user's address space identified by a string found in the struct vidclass vidclasslist[] located in **/etc/conf/pack.o/class.h**.

KDDISPTYPE
This call returns display information to the user. The argument expected is the buffer address of a structure of type *kd_disparam* into which display information is returned to the user. The *kd_disparam* structure is defined as follows:

```
struct kd_disparam {
        long type;        /*display type*/
        char *addr;       /*display memory address*/
        ushort ioaddr[MKDIOADDR];       /*valid I/O addresses*/
}
```

Possible values for the *type* field include:

KD_MONO (0x01), for the IBM monochrome display adapter.

KD_HERCULES (0x02), for the Hercules monochrome graphics adapter.

KD_CGA (0x03), for the IBM color graphics adapter.

KD_EGA (0x04), for the IBM enhanced graphics adapter.

KD_VGA (0x05), for the IBM video graphics adapter.

KDDISPINFO
Returns *struct kd_disparam*, which contains adaptor type and physical address of frame buffer.

KIOCSOUND
Start sound generation. Turn on sound. The *arg* is the frequency desired. A frequency of 0 turns off the sound. This is useful for generating tones while in graphics mode.

KDGETLED
Get keyboard LED status. The argument is a pointer to a character. The character will be filled with a boolean combination of the following values:
1 scroll lock
2 num lock
4 caps lock

KDSETLED
Set keyboard LED status. The argument is a character whose value is the boolean combination of the values listed under "KDGETLED".

KDMKTONE
Not supported. (See KIOCSOUND.)

KDADDIO
Not supported. (See MAP_CLASS.)

KDDELIO
Not supported. (See MAP_CLASS.)

KIOCDOSMODE
Not supported.

KIOCNONDOSMODE
Not supported.

KDSETMODE
(VP/IX only.) Set console in text or graphics mode. The argument is of type integer, which should contain one of the following values:

KD_TEXT	0x00	(sets console to text mode)
KD_GRAPHICS	0x01	(sets console in graphics mode)

Note, the user is responsible for programming the color/graphics adaptor registers for the appropriate graphical state.

KDGETMODE
(VP/IX only.) Get current mode of console. Returns integer argument containing either KD_TEXT or KD_GRAPHICS as defined in the KDSETMODE ioctl description.

KDENABIO
Enable in's and out's to video adaptor ports. No argument.

KDDISABIO
Disable in's and out's to video adaptor ports. No argument.

KDGKBTYPE
Always returns 0.

KDGKBMODE
Get keyborad translation mode, also known as scan code mode. Mode is returned where arg points.

KDSKBMODE
Set keyborad translation mode, also known as scan code mode.

KDGKBSTATE
Returns the state of the shifted, alt-, or control- state of the keyboard. Returns a bollean combination of:
1 shifted
2 control-
4 alt-

KIOCINFO
Always returns 0x6664.

KDMAPDISP
(VP/ix only) Maps display memory into user process address space. Argument is a pointer to structure type *kd_memloc*. This ioctl requires that a virtual 8086 subtask be attached to the current process. KDMAPDISP should not be used by ordinary users to map the console display; use MAPCONS.

KDUNMAPDISP
(VP/ix only) Unmap display memory from user process address space. No argument required.

VT_SETMODE
Set the virtual terminal mode. The argument is a pointer to a vt_mode structure, as defined below.

VT_GETMODE
Determine what mode the active virtual terminal is currently in, either VT_AUTO or VT_PROCESS. The argument to the ioctl is the address of the following type of structure:

```
struct vt_mode {
char mode; /* VT mode */
      char    waitv;  /* not implemented */
      short   relsig; /* signal to use for release request */
      short   acqsig; /* signal to use for display acquired */
      short   frsig;  /* not implemented */
      }

      #define VT_AUTO        0x00    /* automatic VT switching */
      #define VT_PROCESS     0x01    /* process controls switching */
```

The vt_mode structure will be filled in with the current value for each field.

VT_RELDISP
Used to tell the virtual terminal manager that the display has or has not been released by the process.

0 == release refused
1 == release acknowledged
2 == acquire acknowledged

VT_ACTIVATE
Makes the multiscreen number specified in the argument the active multiscreen. The video driver will cause a switch to occur in the same manner as if a hotkey sequence had been typed at the keyboard. If the specified multiscreen is not open or does not exist, the call will fail and errno will be set to ENXIO.

Graphics Adapter Port I/O

You can use *ioctl*(S) to read or write a byte from or to the graphics adapter port. The *arg* parameter of the *ioctl* call uses the *io_arg* data structure:

```
struct port_io_arg {
        struct port_io_struct args[4];
};
```

As shown above, the *io_arg* structure points to an array of four *port_io* data structures. The *port_io* structure has the following format:

```
struct port_io_struct {
        char dir;       /* direction flag (in vs. out) */
        unsigned short port;    /* port address */
        char data;      /* byte of data */
};
```

You may specify one, two, three, or four of the *port_io_struct* structures in the array for one *ioctl* call. The value of *dir* can be either IN_ON_PORT to specify a byte being input to the graphics adapter port or OUT_ON_PORT to specify a byte being output to the graphics adapter port. *Port* is an integer specifying the port address of the desired graphics adapter port. *Data* is the byte of data being input or output as specified by the call.

If you are not using any of the *port_io* structures, load the *port* with 0, and leave the unused structures at the end of the array. Refer to the hardware manuals for port addresses and functions for the various adapters.

You can use the following *ioctl(S)* commands to input or output a byte on the graphics adapter port:

CONSIO	Inputs or outputs a byte on the current graphics adapter port as specified. (All)
MGAIO	Inputs or outputs a byte on the monochrome adapter port as specified. (MONO only)
CGAIO	Inputs or outputs a byte on the color graphics adapter port as specified. (CGA only)
EGAIO	Inputs or outputs a byte on the enhanced graphics adapter port as specified. (EGA only)
VGAIO	Inputs or outputs a byte on the video graphics array adapter port as specified. (VGA only)

To input a byte on any of the graphics adapter ports, load *dir* with IN_ON_PORT and load *port* with the port address of the graphics adapter. The byte input from the graphics adapter port will be returned in *data*.

To output a byte, load *dir* with OUT_ON_PORT, load *port* with the port address of the graphics adapter, and load *data* with the byte you want output to the graphics adapter port.

Function Keys

ioctl(S) can be used to define or obtain the current definition of a function key. The **arg** parameter of the **ioctl** call uses the **fkeyarg** data structure:

```
struct fkeyarg {
        unassigned int keynum;
        char keydef [MAXFK];
        /* Comes from
        char flen; ioctl.h via comcrt.h */
}
```

You can use the following **ioctl**(S) requests to obtain or assign function key definitions:

GETFKEY	Obtains the current definition of a function key. The function key number must be passed in **keynum**. The string currently assigned to the key will be returned in **keydef** and the length of the string will be returned in **flen** when the **ioctl** is performed.

SETFKEY	Assigns a given string to a function key. The function key number must be passed in **keydef** and the length of the string (number of characters) must be passed in **flen**.
SETLOCKLOCK	Toggles the <Caps Lock> and <Num Lock> keys to be either global to all the multiscreens, or local to each individual multiscreen. To make the <Caps Lock> global (its default), set the *arg* parameter to 1. To make the <Caps Lock> local to each screen, set the *arg* parameter to 0.

ANSI Screen Attribute Sequences

The following character sequences are defined by ANSI X3.64-1979 and may be used to control and modify the screen display. Each **n** is replaced by the appropriate ASCII number (decimal) to produce the desired effect. The last column is for *termcap* (M) codes, where "n/a" means not applicable.

The use of 7 or 8 bit characters in the escape sequence is a valid invocation for each action defined. For example the ANSI ED command can be invoked via the "ESC [*n* J" (0x1b-0x5b-*n*-0x4a, 7 bit chars) sequence or the "CSI *n* J" (0x9b-*n*-0x4n, 8 bit chars) sequence.

ISO	Sequence	Action	Termcap Code
ED (Erase in Display)	CSI *n* J	Erases all or part of a display. *n=0*: erases from active position to end of display. *n=1*: erases from the beginning of display to active position. *n=2*: erases entire display.	cd
EL (Erase in Line)	CSI *n* K	Erases all or part of a line. *n=0*: erases from active position to end of line. *n=1*: erases from beginning of line to active position. *n=2*: erases entire line.	ce
ECH (Erase Character)	CSI *n* X	Erases *n* characters	n/a

CBT (Cursor Backward Tabulation)	CSI *n* Z	Moves active position back *n* tab stops.	bt
SU (Scroll Up)	CSI *n* S	Scroll screen up *n* lines, introducing new blank lines at bottom.	sf
SD (Scroll Down)	CSI *n* T	Scrolls screen down *n* lines, introducing new blank lines at top.	sr
CUP (Cursor Position)	CSI m ; *n* H	Moves active position to location *m* (vertical) and *n* (horizontal).	cm
HVP (Horizontal & Vertical Position)	CSI m ; *n* f	Moves active position to location *m* (vertical) and *n* (horizontal).	n/a
CUU (Cursor Up)	CSI *n* A	Moves active position up *n* number of lines.	up (ku)
CUD (Cursor Down)	CSI *n* B	Moves active position down *n* number of lines.	do (kd)
CUF (Cursor Forward)	CSI *n* C	Moves active position *n* spaces to the right.	nd (kr)
CUB (Cursor Backward)	CSI *n* D	Moves active position *n* spaces backward.	bs (kl)
HPA (Horizontal Position Absolute)	CSI *n* ‘	Moves active position to column given by *n*.	n/a
HPR (Horizontal Position Relative)	CSI *n* a	Moves active position *n* characters to the right.	n/a

VPA (Vertical Position Absolute)	CSI *n* d	Moves active position to line given by *n*.	n/a
VPR (Vertical Position Relative)	CSI *n* e	Moves active position down *n* number of lines.	n/a
IL (Insert Line)	CSI *n* L	Inserts *n* new, blank lines.	al
ICH (Insert Character)	CSI *n* @	Inserts *n* blank places for **n** characters.	ic
DL (Delete Line)	CSI *n* M	Deletes *n* lines.	dl
DCH (Delete Character)	CSI *n* P	Deletes *n* number of characters.	dc
CPL (Cursor to Previous Line)	CSI *n* F	Moves active position to beginning of line, *n* lines up.	n/a
CNL (Cursor Next Line)	CSI *n* E	Moves active position to beginning of line, *n* lines down.	n/a

SGR (Select Graphic Rendition)	CSI *n* m	Character attributes, as summarized in the chart below. Multiple attributes can be specified in the form: CSI *n1*; *n2*; *n3* m	n/a

Select Graphic Rendition (SGR) Chart		
n	Meaning	
0	all attributes off (normal display)	
1	bold intensity (or light color)	
4	underscore on (if hardware supports it)	
5	blink on (if hardware supports it)	
7	reverse video	
8	sets blank (non-display)	
10	selects the primary font	
11	selects the first alternate font; lets ASCII characters less than 32 be displayed as ROM characters	
12	selects a second alternate font; toggles high bit of extended ASCII code before displaying as ROM characters	
30	black	foreground
31	red	foreground
32	green	foreground
33	brown	foreground
34	blue	foreground
35	magenta	foreground
36	cyan	foreground
37	white	foreground
38	enables underline option; white foreground with white underscore	
39	disables underline option	
40	black	background
41	red	background
42	green	background
43	brown	background
44	blue	background
45	magenta	background
46	cyan	background
47	white	background

ISO	Sequence	Action	Termcap Code
SM (Set Mode)	CSI [2 h	Lock keyboard. Ignores keyboard input until unlocked. Characters are not saved.	n/a
MC (Media Copy)	CSI [2 i	Send screen to host. Current screen contents are sent to the application.	n/a
RM (Reset Mode)	CSI [2 l	Unlock keyboard. Re-enable keyboard input.	n/a

Additional Screen Attribute Sequences

Name	Sequence	Action	Termcap Code
n/a	CSI = *p*;*d*B	Set the bell parameter to the decimal values of *p* and *d*. *p* is the period of the bell tone in units of 840.3 nanoseconds, and *d* is the duration of the tone in units of 100 milliseconds.	n/a
n/a	CSI = *s*;*e*C	Set the cursor to start on scanline *s* and end on scanline *e*.	n/a
n/a	CSI = *x*D	Turn on or off (*x*=1 or 0) the intensity of the background color.	n/a
n/a	CSI = *x*E	Set or clear (*x*=1 or 0) the Blink vs. Bold background bit in the 6845 crt controller.	n/a
n/a	CSI = *c*A	Set overscan color to color *c*. *c* is a decimal value taken from Color Table above. (This sequence may not be supported on all hardware.)	n/a
n/a	CSI = *c*F	Set normal foreground color to *c*. (*c* is a decimal parameter taken from Color Table.)	n/a

n/a	CSI = *c0G*	Set normal background. (See Color Table.)	n/a
n/a	CSI = *c* H	Set reverse foreground. (See Color Table.)	n/a
n/a	CSI = *c* I	Set reverse background. (See Color Table.)	n/a
n/a	CSI = *c* J	Set graphic foreground. (See Color Table.)	n/a
n/a	CSI = *c* K	Set graphic background. (See Color Table.)	n/a

Color Table			
Cn	Color	Cn	Color
0	Black	8	Grey
1	Blue	9	Lt. Blue
2	Green	10	Lt. Green
3	Cyan	11	Lt. Cyan
4	Red	12	Lt. Red
5	Magenta	13	Lt. Magenta
6	Brown	14	Yellow
7	White	15	Lt. White

Name	Sequence	Action	Termcap Code
n/a	CSI [*n* g	Accesses alternate graphics set. Not the same as "graphics mode." Refer to your owner's manual for decimal/character codes (**Pn**) and possible output characters.	n/a
n/a	ESC Q Fn '*string*'	Define function key **Fn** with *string*. String delimiters ' and ' may be any character not in *string*. **Fn** is defined as the key number starting at zero plus the ASCII value of zero. For example, **F1** = 0... **F16** = ?, and so on. In this escape sequence, the ^ character will cause the next character to have 32 subtracted from its ASCII value. Thus ^! results in a soh (^A) characters.	n/a
n/a	CSI *n* z	*n* should be equal to the number of the screen to switch to. If screen does not exist, no action will take place.	n/a

Files

/dev/console

/dev/tty [02 -*n*]

/dev/color

/dev/monochrome

/dev/ega

/dev/vga

See Also

console(M), ioctl (S), keyboard(HW), keymap(M), mapkey(M), mapchan(M), multiscreen(M), setcolor(C), stty(C), systty(M), vidi(C), termcap(M), tty(M)

scsi

small computer systems interface

Description

The Small Computer Systems Interface (SCSI) provides a standard interface for peripherals such as hard disks, printers, tape drives and others. SCSI is run via a host adapter card that can support up to 7 devices each. The Base I/O Board contains one host adapter (sometimes referred to as "SCSI channel"), whereas High Performance File Processor boards (HPFPs) support up to two host adapters each, with multiple HPFPs supported as well.

The minor device numbering scheme for SCSI disk devices is the same as the standard minor device number scheme for non-SCSI disk devices, with the exception that multiple major numbers are allowed (up to 16 contiguous major numbers).

The minor device numbering scheme for SCSI tape devices is as follows:

SCSI Tape Minor Devices

Bits								
7	6	5	4	3	2	1	0	Description
X	-	-	-	-	-	-	-	Override device
-	X	X	-	-	-	-	-	Data density
-	-	-	X	-	-	-	-	Variable length
-	-	-	-	X	-	-	-	No rewind on close
-	-	-	-	-	X	X	X	Unit number

See Also

hd(HW), tape(HW), *System Administrator's Guide*

Value Added

scsi is an extension of AT&T System V provided in Altos UNIX System V.

serial: tty1[a-h] , tty1[A-H] , tty2[a-h] , tty2[A-H]

interface to serial ports

Description

This section describes the COM1 and COM2 serial ports. For information on the serial ports available with the Multidrop (MDC/2) or the Serial Concentrator (SIO/2) boards, refer to the appropriate hardware manual accompanying appropriate to your configuration.

The **tty1[a-h]**, **tty1[A-H]**, **tty2[a-h]** and **tty2[A-H]** files provide access to the standard and optional serial ports of the computer. Each file corresponds to one of the serial ports (with or without modem control). Files are named according to the following conventions:

- The first number in the file name corresponds to the COM expansion slot.
- Lower case letters indicate no modem control.
- Upper case letters indicate the line has modem control.

tty1a and **tty1A** both refer to COM 1, whereas **tty2a** and **tty2A** both refer to COM 2.

The COM1 and COM2 ports each have modem and non-modem invocations. The device names in the following table refer to the serial ports, with and without modem control. The first section of the table describes boards at COM 1 and the second section describes boards installed at COM 2. "Minor" is the minor device number for the port (see *mknod*(C)).

Serial Lines					
Board Type		Non-Modem Control		Modem Control	
		Minor	*Name*	*Minor*	*Name*
	1 Port	0	tty1a	128	tty1A
	4 Port	1	tty1b	129	tty1B
		2	tty1c	130	tty1C
		3	tty1d	131	tty1D
8 Port		4	tty1e	132	tty1E
		5	tty1f	133	tty1F
		6	tty1g	134	tty1G
		7	tty1h	135	tty1H
	1 Port	8	tty2a	136	tty2A
	4 Port	9	tty2b	137	tty2B
		10	tty2c	138	tty2C
		11	tty2d	139	tty2D
8 Port		12	tty2e	140	tty2E
		13	tty2f	141	tty2F
		14	tty2g	142	tty2G
		15	tty2h	143	tty2H

Interrupt Vectors:

All board(s) installed at COM 1 - 4
All board(s) installed at COM 2 - 4

For a list of I/O addresses, see the *Release Notes* furnished with your distribution.

Access

The files may only be accessed if the corresponding serial interface card is installed and its jumper I/O address correctly set. Also, for multi-port expansion cards, you must use the *mkdev*(ADM) program to create more than the default number of files. Unless other COM slots are specifically referred to in your hardware documentation, only COM 1 and COM 2 may be used.

The serial ports must also be defined in the system configuration. Check your hardware manual to determine how your system is configured, via a CMOS database or by switch settings on the main system board. If your system is configured using a CMOS database, the ports are defined in the database (see *cmos*(HW)). Otherwise, define the ports by setting the proper switches on the main system board. Refer to your computer hardware manual for switch settings.

It is an error to attempt to access a serial port that has not been installed and defined.

The serial ports can be used for a variety of serial communication purposes such as connecting login terminals to the computer, attaching printers, or forming a serial network with other computers. Note that a serial port may operate at most of the standard baud rates, and that the ports (on most computers) have a DTE (Data Terminal Equipment) configuration. The following table defines how each pin is used for 25-pin and 9-pin connections:

25-Pin	9-Pin	Description
2	2	Transmit Data
3	3	Receive Data
4	7	Request to Send
5	8	Clear to Send
7	5	Signal Ground
8	1	Carrier Detect (Data Set Ready)
20	4	Data Terminal Ready

Only pins 2, 3, and 7 (2,3 and 5 for 9-pin) are necessary for a terminal (or direct) connection.

A modem control device (port) uses pins 2, 3, and 7 in the same way as a non-modem control device: send on pin 2 and receive on pin 3. Pin 7 is data ground. On a non-modem control device, pins 4 and 20 (RTS and DTR) are asserted, but pin 8 is not. On a modem control device, pins 4 and 20 (RTS & DTR) are asserted and the port will not open until pin 8 (CXD) is asserted. That is, no signal travels from pin 2 until pin 8 is asserted from another source. The modem control device monitors the the status of pin 8.

See *tty*(M) and *termio*(M) for the details of serial line operation on UNIX systems.

Files

/dev/tty1[a-h]
/dev/tty1[A-H]
/dev/tty2[a-h]
/dev/tty2[A-H]

See Also

cmos(HW), csh(C), cu(C), getty(ADM), mkdev(ADM), mknod(C) nohup(C), open(S), termio(M), tty(M), uucp(C)

Notes

If you login via a modem control serial line, hanging up logs that line out and kills your background processes. See *nohup*(C) and *csh*(C)).

You cannot use the same serial port with both modem and non-modem control at the same time. For example, you cannot use tty1a and tty1A simultaneously.

Use a modem cable to connect your modem to a computer.

tape

magnetic tape device

Description

The *tape* device implements the UNIX interface with a tape drive. QIC-02 cartridge tape drives are supported by the *ct* device driver, QIC-40 and QIC-80 tape drives connected to the floppy disk controller are supported with the *ft* device driver, and Irwin tape drives connected to the floppy disk controller are supported with the *ir* device driver.

Typically, the *tar*(C), *cpio*(C), *dd*(C), *backup*(ADM), *xbackup*(ADM), *xrestore*(ADM), or *restore*(ADM) commands are used to access a tape drive.

A single tape drive with a raw (character, non-blocking) interface is supported, except for the SCSI tape driver which supports up to eight devices.

There are numerous "types" offered for each tape device, resulting in numerous device files for a single tape device. (See "Files" section.) The general naming format for common tape device types is shown below:

/dev/[n][v][l|m|h][r][Stp|ct|ft|ir]*x*

where:

n	no rewind	**r**	raw device
v	variable block size	**Stp**	SCSI tape device
l	low-density format	**ct**	cartridge tape device (QIC-02)
m	medium-density format	**ft**	tape attached to floppy controller
h	high-density format	**ir**	Irwin tape (floppy controller)
		x	the unique tape device number

Other tape types include these special prefixes:

e	ECC error recovery device	x	special control device

These device types and their filename codes are described in the following paragraphs.

Currently, eight tape devices are supported, numbered 0 through 7. For example, the first raw cartridge tape device is **/dev/rct0**.

Devices beginning with the "r" prefix, (for "raw device"), should be used for most normal tape work, while devices with the "n" prefix, ("for no rewind on hold"), should be used for storing and restoring multiple files. Devices beginning with the "x" prefix are control devices, which are used for sending *ioctl*(S) commands to the tape subsystem.

For SCSI 9-track tapes, there will be two variable length devices: **vrStp***x* and **nvrStp***x* (for the no-rewind device). They will always be created when a SCSI tape drive is added, but they can only be used with 9-track devices that support variable length.

SCSI 9-track and Exabyte tape drives should be added with **mkdev tape** and by choosing the SCSI options.

Devices beginning with the "e" prefix (for ECC device) support a 2/64 error recovery scheme. Thus two 512-byte blocks out of every 64 blocks can be bad and the driver will correct the errors. This software ECC support provides a high degree of error recovery.

The *ft* and *ir* floppy tape drivers do not support the "n" or "e" device types. ECC encoding and decoding is automatically used with the standard "r" device. On the QIC-40, QIC-80 and Irwin 80MB drives, for every 29K written to the tape, 3K of ECC data is written with it to provide error recovery. On the Irwin 10, 20, 40 and 60MB drives, for every 16K written to the tape, 2K of ECC data is written.

QIC-40 and QIC-80 tapes must be formatted with the *tape*(C) command before use, unless you use pre-formatted tapes. Similarly, Irwin tapes must be first servo-written and then formatted with *tape*(C) before use, unless you use pre-formatted tapes. The *ir* driver can read tapes formatted and written under UNIX but cannot write to them.

The density-format prefixes ''l'', ''m'', and ''n'', force the specified tape device to read and write data in ''low'', ''medium'', or ''high'' density format, regardless of the drive's auto-sensing capabilities. Thus, a Tandberg tape drive will attempt to write in qic-320 format (high-density) if a DC 6525 tape is inserted and the drive is allowed to choose a format. By using the medium-density device file, the user can cause the drive to write in qic-150 format, so that the resulting tape is readable by devices that are capable of reading only qic-150 format tapes.

Another, less obvious, use of forcing drives to read at a specified density enables you to determine the density of an unknown tape. To determine what density a tape has been written in, try to read the tape at the three different density devices for your tape drive. The device that successfully reads the tape indicates the tape density.

Tape device filenames without any of these three density-specifying prefixes indicates a "default-density" device, in which the tape device sets the density to its default value.

The following table shows how specifying density via the device filename prefix (l, m, h, or default) affects the tape format used by various brands of tape drives:

DENSITY	MANUFACTURER			
	Tandberg	Archive	Wangtek	Qualstar
Default	Auto-Senses	Auto-senses	Auto-senses	6250 bpi †
Low (l)	QIC-120	QIC-24*	QIC-24*	1600 bpi
Medium (m)	QIC-150	QIC-120	QIC-120	3200 bpi
High (h)	QIC-320	QIC-150	QIC-150	6250 bpi

* Read only.
† Auto-senses only on read.

Note that when auto-sensing by default (as shown above), a tape device writes at the highest density possible. However, when auto-sensing during a read, the drive will read at any density within its repertoire, whichever approrpiate. Also note that a drive may be capable of reading or writing in a density not listed in the above table. Except as noted, any drive not listed above will only use its default density, regardless of it full capabilities.

Exabyte drives are only capable of reading and writing a single density. This density is not defined by the ANSI SCSI specification, and so varies from model to model.

The table below shows the maximum density (format) possible for selected tape sizes.

TAPE SIZE	MAX. DENSITY
6526	QIC-320
6150	QIC-150
600XTD	QIC-150
600A	QIC-120
300XLP	QIC-24

The following table summarizes the base naming conventions for the tape drives supported:

ct0,1	QIC24 unit 0,1
ct2,3	QIC11 unit 0,1
Stp0,1,2,3,4,5,6,7	SCSI tape unit 0,1,2,3,4,5,6,7
ft0	QIC-40 or QIC-80 floppy tape unit
ir0	Irwin floppy tape unit
ctmini	default mini-cartridge device

The default tape device is stored in the file **/etc/default/tape**, which is also used by *tape*(C). **/etc/default/tape** should always contain the "x" (control) device name of the default device, and is normally updated by the **mkdev tape** command. If the default device is an QIC-40, QIC-80 or Irwin tape drive, the appropriate device from the table above will be linked to the *ctmini* device node. QIC-02 tape drives will always be accessed by the *ct0,1* device nodes as shown in the table. If a SCSI tape drive is installed as the default device and there is no QIC-02 drive installed, it will be linked to the *ct0* device node. If both SCSI and QIC-02 drives are installed, the SCSI device node cannot be linked to the *ct0* device node.

tape(C) describes the commands used to access tape drives.

Definition of ioctl commands

The following *ioctl* commands can be used with the various tape device drivers supported under UNIX. The letters following each description indicate which drivers support each *ioctl* command:

A	All drivers
C	QIC-02 cartridge tape driver
S	SCSI tape driver
F	QIC-40 and QIC-80 mini-cartridge tape drivers
I	Irwin mini-cartridge tape driver

MT_STATUS
Returns a device-independent structure holding the status of the drive. The *tape_info* structure is defined in **/usr/include/sys/tape.h.** (A)

MT_DSTATUS
Returns a device-dependent structure holding status information of the drive. If this command does not seem to work, use stp_status(dev) instead. (A)

MT_RESET
Resets the driver software and the tape drive. Interrupts tape commands in progress. (C,F,I)

MT_REPORT
Returns an integer code which determines the type of device which the driver controls. The type numbers are defined in **/usr/include/sys/tape.h.** (A)

MT_RETEN
Winds the tape forward to EOT and then backward to BOT. (A)

MT_REWIND
Rewinds the tape to BOT. (A)

MT_ERASE
Erases the data on the tape and retensions the cartridge. (C,S,F)

MT_AMOUNT
Returns an integer count of the amount of the last data transfer. This command will fail if there is no tape in the drive. (A)

MT_FORMAT
Formats the tape. Expects as an argument the number of tracks to format, which must be an even number. If no argument is provided, the default is 20 tracks for QIC-40 drives, 28 tracks for QIC-80 drives, and from 8 to 32 tracks for Irwin drives, depending on the capacity of the tape. On Irwin drives, the tape must previously have been servo-written before formatting, either by the manufacturer or with the MT_SERVO command. (F,I)

MT_GETHDR
Expects as an argument a pointer to a *struct ft_header* or *struct ir_header* and copies the header of the current tape into it. (F,I)

MT_PUTHDR
Takes a pointer to a *struct ft_header* or *struct ir_header* and writes it onto the tape. This command should be used with caution. (F,I)

MT_GETNEWBB
Takes a pointer to a *struct ft_newbbt* or *struct ir_newbbt* and copies in a list of bad blocks detected on the last write operation. (F,I)

MT_PUTNEWBB
Takes a pointer to a *struct ft_newbbt* or *struct ir_newbbt*, reads in the header from the tape, then writes a new bad block onto the tape with the new bad blocks from the provided bad block table. (F,I)

MT_GETVTBL
Takes a pointer to a *struct ft_vtbl* and copies in the volume table from the tape. (F)

MT_PUTVTBL
Takes a pointer to a *struct ft_vtbl* and writes the volume table onto the tape. This command should be used with caution. (F)

MT_SERVO
Writes servo marks on a blank tape in preparation for formatting with MT_FORMAT. If the tape has previously been servo-written, this command may fail unless the tape is first bulk-erased with a commercial tape eraser. Normally, a tape should only be servo-written once in its lifetime, although it can be formatted many times. (I)

MT_RFM
Winds the tape forward to the next file mark. (C,S)

MT_WFM
Writes a file mark at the current location on the tape. (C,S)

MT_LOAD
On devices which are capable of doing so, loads the tape into the drive. (S)

MT_UNLOAD
On devices which are capable of doing so, unloads the tape from the drive. (S)

Files

For a common SCSI tape drive:

/dev/hrStp*x*	/dev/nmrStp*x*	/dev/nvrStp*x*	/dev/vrStp*x*
/dev/lrStp*x*	/dev/nrStp*x*	/dev/rStp*x*	/dev/xStp*x*
/dev/mrStp*x*	/dev/nvhrStp*x*	/dev/vhrStp*x*	
/dev/nhrStp*x*	/dev/nvlrStp*x*	/dev/vlrStp*x*	
/dev/nlrStp*x*	/dev/nvmrStp*x*	/dev/vmrStp*x*	

A selection of other tape drives:

/dev/xft*x*	/dev/rir*x*	/dev/erct*x*	/dev/rctmini
/dev/rft*x*	/dev/nrct*x*	/dev/xct*x*	
/dev/xir*x*	/dev/rct*x*	/dev/xctmini	

Include files:

/usr/include/sys/tape.h
/usr/include/sys/ct.h
/usr/include/sys/ft.h
/usr/include/sys/ir.h

Notes

After certain tape operations are executed, the system returns a prompt before the tape controller has finished its operation. If the user enters another tape command too quickly, a "device busy" error is returned until the tape device is finished with its previous operation.

Periodic tape cartridge retensioning and tape head cleaning are necessary for continued error-free operation of the tape subsystem. Use *tape*(C) to retension the tape.

See Also

backup(ADM), xbackup(ADM), cpio(C), dd(C), format(C), tape(C), tar(C), restore(ADM), xrestore(ADM)

terminal

login terminal

Description

A *terminal* is any device used to enter and display data. It may be connected to the computer:

- By a serial wire, either direct or dialup
- As a virtual terminal, for example with emulator software
- Through a display adapter

A terminal has an associated device file **/dev/tty***.

Set up ports for use by terminals with the **pcu**(ADM) command.

Files

/dev/tty*

See Also

console(M), disable(C), enable(C), mkdev(ADM), serial(HW), stty(C), vidi(C), termcap(M), term(F), terminals(M), pcu(ADM)

xt

multiplexed tty driver for AT&T windowing terminals

Description

The *xt* driver provides virtual *tty*(M) circuits multiplexed onto real *tty*(M) lines. It interposes its own channel multiplexing protocol as a line discipline between the real device driver and the standard *tty*(M) line disciplines.

Virtual *tty*(M) circuits are named by character-special files of the form */dev/xt???*. File names end in three digits, where the first two represent the channel group and the last represents the virtual *tty*(M) number (0-7) of the channel group. Allocation of a new channel group is done dynamically by attempting to open a name ending in **0** with the **O_EXCL** flag set. After a successful open, the *tty*(M) file onto which the channels are to be multiplexed should be passed to *xt* via the **XTIOCLINK** *ioctl*(S) request. Afterwards, all the channels in the group will behave as normal *tty*(M) files, with data passed in packets via the real *tty*(M) line.

The *xt* driver implements the protocol described in *xtproto*(M) and in *layers*(M). Packets are formatted as described in *xtproto*(M), while the contents of packets conform to the description in *layers*(M).

There are three groups of *ioctl*(S) requests recognized by *xt*. The first group contains all the normal tty *ioctl*(S) requests described in *termio*(M), with the addition of the following:

TIOCEXCL Set exclusive use mode; no further opens are permitted until the file has been closed.

TIOCNXCL Reset exclusive use mode; further opens are once again permitted.

The second group of *ioctl*(S) requests concerns control of the windowing terminal, and is described in the header file **<sys/jioctl.h>**. The requests are as follows:

JTYPE, JMPX Both return the value **JMPX.** These are used to identify a terminal device as an *xt* channel.

JBOOT, JTERM Both generate an appropriate command packet to the windowing terminal affecting the layer associated with the file descriptor argument to *ioctl*(S). They may return the error code **EIO** if the system *clist* is empty.

JTIMO, JTIMOM — **JTIMO** specifies the timeouts in seconds, and **JTIMOM** in milliseconds. Invalid except on channel 0. They may return the error code **EIO** if the system *clist* is empty.

JWINSIZE — Requires the address of a *jwinsize* structure as an argument. The window sizes of the layer associated with the file descriptor argument to *ioctl*(S) are copied to the structure.

JZOMBOOT — Generate a command packet to the windowing terminal to enter download mode on the channel associated with the file descriptor argument to *ioctl*(S), like **JBOOT**; but when the download is finished, make the layer a zombie (ready for debugging). It may return the error code **EIO** if the system *clist* is empty.

JAGENT — Send the supplied data as a command packet to invoke a windowing terminal agent routine, and return the terminal's response to the calling process. Invalid except on the file descriptor for channel 0. See *jagent*(M). It may return the error code **EIO** if the system *clist* is empty.

The third group of *ioctl*(S) requests concerns the configuration of *xt*, and is described in the header file <sys/xt.h>. The requests are as follows:

XTIOCTYPE — Returns the value **XTIOCTYPE.**

XTIOCLINK — Requires an argument that is a structure, *xtioclm*, containing a file descriptor for the file to be multiplexed and the maximum number of channels allowed. Invalid except on channel 0. This request may return one of the following errors:

- **EINVAL** — *nchans* has an illegal value.
- **ENOTTY** — *fd* does not describe a real *tty*(M) device.
- **ENXIO** — *linesw* is not configured with *xt*.
- **EBUSY** — An **XTIOCLINK** request has already been issued for the channel group.
- **ENOMEM** — There is no system memory available for allocating to the *tty*(HW) structures.

	EIO	The **JTIMOM** packet described above could not be delivered.

HXTIOCLINK Like **XTIOCLINK**, but specifies that ENCODING MODE be used.

XTIOCTRACE Requires the address of a *tbuf* structure as an argument. The structure is filled with the contents of the driver trace buffer. Tracing is enabled. This request is invalid if tracing is not configured.

XTIOCNOTRACE Tracing is disabled. This request is invalid if tracing is not configured.

XTIOCSTATS Requires an argument that is the address of an array of size S_NSTATS, of type *Stats_t*. The array is filled with the contents of the driver statistics array. This request is invalid if statistics are not configured.

XTIOCDATA Requires the address of a maximum-sized *Link* structure as an argument. The structure is filled with the contents of the driver *Link* data. This request is invalid if data extraction is not configured.

Files

/dev/xt/??[0-7]	multiplexed special files
/usr/include/sys/jioctl.h	packet command types
/usr/include/sys/xtproto.h	channel multiplexing protocol definitions
/usr/include/sys/xt.h	driver specific definitions

See Also

layers(C), termio(M), tty(M), ioctl(S), open(S), libwindows(S), jagent(M), layers(M)

Permuted Index

Commands, System Calls, Library Routines and File Formats

This permuted index is derived from the "Name" description lines found on each reference manual page. Each *index* line shows the title of the entry to which the line refers, followed by the reference manual section letter where the page is found.

To use the *permuted index* search the middle column for a key word or phrase. The right hand column contains the name and section letter of the manual page that documents the key word or phrase. The left column contains additional useful information about the command. Commands or routines are also listed in the context of the *index* line, followed by a colon (:). This denotes the "beginning" of the sentence. Notice that in many cases, the lines wrap, starting in the middle column and ending in the left column. A slash (/) indicates that the description line is truncated.

Change Information

CH

This is a summary of the changes that have been made to the previous version of this manual. The chapters, page numbers, and/or paragraphs mentioned in this summary refer to the previous manual.

Title: Altos UNIX System V/386 Release 3.2
System Administrator's Reference (ADM,HW)

Revised Part Number: 690-23416-002

Previous Part Numbers: 690-23416-001 and 690-24539-001

Date: March 1991

Changes:

The phrase, "for the 486" (or in some manuals "for Entry-level Systems") was deleted to indicate that this operating system runs on a wider range of platforms.

The "Guide to Your Altos UNIX System V/386 Release 3.2 Documentation" and the "Operating System Documents for Different Audiences" pages located in the front matter of the manual were both changed to indicate that a different combination of the available manuals are now shipped with every run-time system. This has also affected which manuals are now available as spare parts.

Added *add.vd*(ADM), *addxusers*(ADM), *checkaddr*(ADM), *checkup*(ADM), *del.vd*(ADM), *dlayout*(ADM), *hdutil*(ADM), *idaddld*(ADM), *idcheck*(ADM), *idmemtune*(ADM), *initscr*(ADM), *strmcfg*(ADM), *strmtune*(ADM), *tcbck*(ADM), *upsconfig*(ADM), *vddaemon*(ADM), *xprsetup*(ADM), and *cdrom*(HW).

Deleted *sfmt*(ADM), *vdsetup*(ADM), *vdshut*(ADM).

Miscellaneous editing and typographical changes were made throughout the manual.

Page	Command	Description
1	add.vd(ADM)	No longer asks you for stripe size or default number of files to use. This command is normally not invoked directly, since the new *vdutil*(ADM) command invokes this command.
1	configure(ADM)	Added **-o** option. Also included small changes to main menu display.
1	crash(ADM)	Added **-r** and **-o** options, to provide support for UPS shutsave requirements.
1,2	divvy(ADM)	Made it clear that the **-s** option takes as its argument the SCSI index number. Also emphasized that the **-v** option takes as its argument a partition number, and that it has nothing to do with virtual disks. The system now supports 52 disks (SCSI index numbers **0** to **51**, and disk letters **a** to **z** and **A** to **Z**).
7	divvy(ADM)	Also, changed the table in the "SCSI Conversion" section to indicate new device number allocation scheme. Device numbers are allocated on a first-come, first-served basis.
2	fsck(ADM)	The **-rr** option is used to recover XENIX root hard disks. Added the **-b** option, which preforms the same operation for UNIX root filesystems.
2,3	fdisk(ADM)	Fixed incorrect hard disk device filename. Also, minimum recommended size for a UNIX partition is now 50 MB, not 40 MB.
1-3	hdutil(ADM)	The **-c** option to move root hard disk device names is deleted. Deleted the "Moving the Root Disk" section. Also increased range of hard disk letters from **W** up to **Z**.

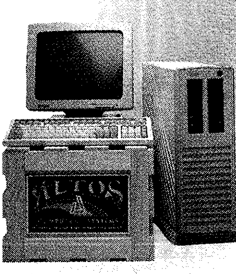
Page	Command	Description
1,3	mkdev(ADM)	Added *mkdev graphics* option, which some systems might not have had documented. Emphasized that *mkdev hd* does *not* support other controllers besides SCSI, and that *badtrk* is not invoked. Made other miscellaneous *mkdev hd* corrrections.
all	pcu(ADM)	Added **-x** option, for attaching transparent printers. Summarized the sequence of steps to follow when adding a new tty controller to the system. All screen labels now have only one capital letter, so that they can be invoked with a single keystroke (function-key substitution). Also described keystroke substitutes for arrow-keys. Emphasized that the **-d** and **-a** options require a space between the option and their arguments. Disabled ports are displayed with a colon (:) preceding their name. Described changes to the **card_info** file that affect *pcu*, including the addition of a new field and distinctions between ISA and EISA cards.
2	scsinfo(ADM)	Logical SCSI index numbers for hard disks now range from 0 to 51, not 34. Also, logical index numbers for SCSI tapes range only from 0 to 7, not up to 15. Deleted numbering information found in the ''Notes'' section that no longer applies to the current scheme.
1,2	strmcfg(ADM)	Added the **-p** option. Added another configurable parameter, NSTREAM. Described data files in the directory, **./config.**
1	swap(ADM)	Corrected erroneous hard disk device filename.

Page	Command	Description
1,2	uconfig(ADM)	*uconfig* now sets up terminal database for system security services. Added **-t** option, to suppress setting up protected terminal database. Made clear all EISA-specific tasks.
1-2	upsconfig(ADM)	Added **-i**, **-t**, and **-d** options. Described the added support for UPS shutsave operation.
1	vddaemon(ADM)	Deleted all options; they are no longer accepted. *vddaemon* is always started (by *init*), thus no longer requiring a manual invocation. *vdshut* has been deleted.
1-2	vdinfo(ADM)	Support for both mirror and striped disks has been added, for those systems that did not have full functionality before. Corrected hard disk device filename example. Added more examples of typical *vdinfo* output.
1-4	vdutil(ADM)	Numerous changes were made to the *vdutil* functionality and appearance. *vdutil* is now the preferred utility for all virtual disk operations.
2-4	boot(HW)	Changed the format of the standalone boot program, including the addition of new parameters. Described how these bootstring parameters change meaning depending on the exact number of parameters used. Also changed the format of additional arguments to the bootstring. Added new aliasing information.
4	boot(HW)	The **scrn** value in the bootstring **systty** is **cn** for the display adapter, not **scrn**.

CH

Page	Command	Description
5-6	boot(HW)	Replaced old aliasing example with a better one. Added two new boot options, **RESTART** and **RSPART**, to provide boot support for UPS shutsave operations. Also added two new command line parameters, **verbose** and **restart**.
1-4	hd(HW)	Simplified the format description of hard disk device filenames. The range of valid hard disk ''numbers'' has increased from **W** to **Z**. Major numbers for hard disks can now range up to 76, not 73. Deleted descriptions of old allocation schemes.

P/N 690-23416-002
Printed in U.S.A.
7/91



Altos Computer Systems
2641 Orchard Parkway, San Jose, CA 95134
408/432-6200, FAX 408/435-8517