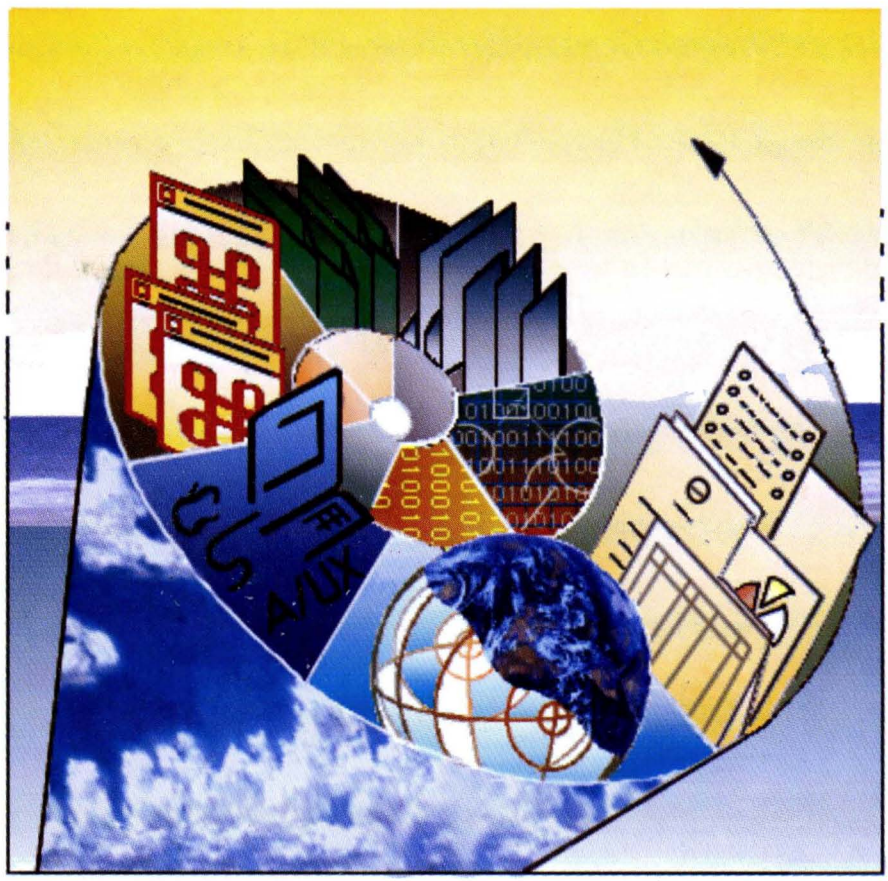
A/UX® System Administrator's Reference
Sections 1M, 7, and 8



A/UX® System Administrator's Reference

Sections 1M, 7, and 8

030-0779

APPLE COMPUTER, INC.

Apple Computer, Inc.
20525 Mariani Ave.
Cupertino, California 95014
(408) 996-1010

Simultaneously published in the United States and Canada.

030-0779

LIMITED WARRANTY ON MEDIA AND REPLACEMENT

If you discover physical defects in the manual or in the media on which a software product is distributed, Apple will replace the media or manual at no charge to you provided you return the item to be replaced with proof of purchase to Apple or an authorized Apple dealer during the 90-day period after you purchased the software. In addition, Apple will replace damaged software media and manuals for as long as the software product is included in Apple's Media Exchange Program. While not an upgrade or update method, this program offers additional protection for up to two years or more from the date of your original purchase. See your authorized Apple dealer for program coverage and details. In some countries the replacement period may be different, check with your authorized Apple dealer.

ALL IMPLIED WARRANTIES ON THIS MANUAL, INCLUDING IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, ARE LIMITED IN DURATION TO NINETY (90) DAYS FROM THE DATE OF THE ORIGINAL RETAIL PURCHASE OF THIS PRODUCT.

Even though Apple has reviewed this manual, **APPLE MAKES NO WARRANTY OR REPRESENTATION, EITHER EXPRESS OR IMPLIED, WITH RESPECT TO THIS MANUAL, ITS QUALITY, ACCURACY, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. AS A RESULT, THIS MANUAL IS SOLD "AS IS," AND YOU, THE PURCHASER, ARE ASSUMING THE ENTIRE RISK AS TO ITS QUALITY AND ACCURACY.**

IN NO EVENT WILL APPLE BE LIABLE FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES RESULTING FROM ANY DEFECT OR INACCURACY IN THIS MANUAL, even if advised of the possibility of such damages.

THE WARRANTY AND REMEDIES SET FORTH ABOVE ARE EXCLUSIVE AND IN LIEU OF ALL OTHERS, ORAL OR WRITTEN, EXPRESS OR IMPLIED. No Apple dealer, agent, or employee is authorized to make any modification, extension, or addition to this warranty.

Some states do not allow the exclusion or limitation of implied warranties or liability for incidental or consequential damages, so the above limitation or exclusion may not apply to you. This warranty gives you specific legal rights, and you may also have other rights which vary from state to state.

A/UX System Administrator's Reference

Contents

Preface

Conventions Used in This Manual

A/UX® manuals follow certain conventions regarding presentation of information. Words or terms that require special emphasis appear in specific fonts within the text of the manual. The following sections explain the conventions used in this manual.

Significant fonts

Words that you see on the screen or that you must type exactly as shown appear in `Courier` font. For example, when you begin an A/UX work session, you see the following on the screen:

```
login:
```

The text shows `login:` in `Courier` typeface to indicate that it appears on the screen. If the next step in the manual is

Enter `start`

`start` appears in `Courier` to indicate that you must type in the word. Words that you must replace with a value appropriate to a particular set of circumstances appear in *italics*. Using the example just described, if the next step in the manual is

`login:` *username*

you type in your name—`Laura`, for example— so the screen shows:

```
login: Laura
```

Key presses

Certain keys are identified with names on the keyboard. These modifier and character keys perform functions, often in combination with other keys. In the manuals, the names of these keys appear in the format of an Initial Capital letter followed by SMALL CAPITAL letters.

The list that follows provides the most common keynames.

RETURN	DELETE	SHIFT	ESCAPE
OPTION	CAPS LOCK	CONTROL	

For example, if you enter

```
Applee
```

instead of

```
Apple
```

you would position the cursor to the right of the word and press the DELETE key once to erase the additional *e*.

For cases in which you use two or more keys together to perform a specific function, the keynames are shown connected with hyphens. For example, if you see

Press CONTROL-C

you must press CONTROL and C simultaneously (CONTROL-C normally cancels the execution of the current command).

Terminology

In A/UX manuals, a certain term can represent a specific set of actions. For example, the word *Enter* indicates that you type in an entry and press the RETURN key. If you were to see

Enter the following command: `whoami`

you would type `whoami` and press the RETURN key. The system would then respond by identifying your login name.

Here is a list of common terms and their corresponding actions.

Term	Action
Enter	Type in the entry and press the RETURN key
Press	Press a *single* letter or key *without* pressing the RETURN key
Type	Type in the letter or letters *without* pressing the RETURN key
Click	Press and then immediately release the mouse button

Term	Action
Select	Position the pointer on an item and click the mouse button
Drag	Position the pointer on an icon, press and hold down the mouse button while moving the mouse. Release the mouse button when you reach the desired position.
Choose	Activate a command title in the menu bar. While holding down the mouse button, drag the pointer to a command name in the menu and then release the mouse button. An example is to drag the File menu down until the command name Open appears highlighted and then release the mouse button.

Syntax notation

A/UX commands follow a specific order of entry. A typical A/UX command has this form:

`command` [*flag-option*] [*argument*] . . .

The elements of a command have the following meanings.

Element	Description
`command`	Is the command name.
flag-option	Is one or more optional arguments that modify the command. Most flag-options have the form [-opt...] where opt is a letter representing an option. Commands can take one or more options.
argument	Is a modification or specification of the command; usually a filename or symbols representing one or more filenames.

Element	Description
brackets ([ ])	Surround an optional item—that is, an item that you do not need to include for the command to execute.
ellipses (...)	Follow an argument that may be repeated any number of times.

For example, the command to list the contents of a directory (`ls`) is followed below by its possible flag options and the optional argument *names*.

```
ls [-R] [-a] [-d] [-C] [-x] [-m] [-l] [-L]
[-n] [-o] [-g] [-r] [-t] [-u] [-c] [-p] [-F]
[-b] [-q] [-i] [-s] [names]
```

You can enter

```
ls -a /users
```

to list all entries of the directory `/users`, where

`ls`	Represents the command name
`-a`	Indicates that *all* entries of the directory be listed
`/users`	Names which directory is to be listed

Command reference notation

Reference material is organized by section numbers. The standard A/UX cross-reference notation is

cmd(sect)

where *cmd* is the name of the command, file, or other facility; *sect* is the section number where the entry resides.

- ☐ Commands followed by section numbers (1M), (7), or (8) are listed in *A/UX System Administrator's Reference*.
- ☐ Commands followed by section numbers (1), (1C), (1G), (1N), and (6) are listed in *A/UX Command Reference*.
- ☐ Commands followed by section numbers (2), (3), (4), and (5) are listed in *A/UX Programmer's Reference*.

For example,

```
cat(1)
```

refers to the command `cat`, which is described in Section 1 of *A/UX Command Reference*. References can also be called up on the screen. The `man` command or the `apropos` command displays pages from the reference manuals directly on the screen. For example, enter the command

```
man cat
```

In this example, the manual page for the `cat` command including its description, syntax, options, and other pertinent information appears on the screen. To exit, continue pressing the space bar until you see a command prompt, or press Q at any time to return immediately to your command prompt. The manuals often refer to information discussed in another guide in the suite. The format for this type of cross reference is "Chapter Title," *Name of Guide*. For a complete description of A/UX guides, see *Road Map to A/UX Documentation*. This guide contains descriptions of each A/UX guide, the part numbers, and the ordering information for all the guides in the A/UX documentation suite.

Introduction

to the A/UX Reference Manuals

1. How to use the reference manuals

A/UX Command Reference, *A/UX Programmer's Reference*, and *A/UX System Administrator's Reference* are reference manuals for all the programs, utilities, and standard file formats included with your A/UX® system.

The reference manuals constitute a compact encyclopedia of A/UX information. They are not intended to be tutorials or learning guides. If you are new to A/UX or are unfamiliar with a specific functional area (such as the shells or the text formatting programs), you should first read *A/UX Essentials* and the other A/UX user guides. After you have worked with A/UX, the reference manuals help you understand new features or refresh your memory about command features you already know.

2. Information contained in the reference manuals

A/UX reference manuals are divided into three volumes:

- The two-part *A/UX Command Reference* contains information for the general user. It describes commands you type at the A/UX prompt that list your files, compile programs, format text, change your shell, and so on. It also includes programs used in scripts and command language procedures. The commands in this manual generally reside in the directories `/bin`, `/usr/bin` and `/usr/ucb`.
- The two-part *A/UX Programmer's Reference* contains information for the programmer. It describes utilities for programming, such as system calls, file formats of subroutines, and miscellaneous programming facilities.
- *A/UX System Administrator's Reference* contains information for the system administrator. It describes commands you type at the A/UX prompt to control your machine, such as accounting

commands, backing up your system, and charting your system's activity. These commands generally reside in the directories `/etc`, `/usr/etc`, and `/usr/lib`.

These areas can overlap. For example, if you are the only person using your machine, then you are both the general user and the system administrator.

To help direct you to the correct manual, you may refer to *A/UX Reference Summary and Index*, which is a separate volume. This manual summarizes information contained in the other A/UX reference manuals. The three parts of this manual are a classification of commands by function, a listing of command synopses, and an index.

3. How the reference manuals are organized

All manual pages are grouped by section. The sections are grouped by general function and are numbered according to standard conventions as follows:

1	User commands
1M	System maintenance commands
2	System calls
3	Subroutines
4	File formats
5	Miscellaneous facilities
6	Games
7	Drivers and interfaces for devices
8	A/UX Startup shell commands

Manual pages are collated alphabetically by the primary name associated with each. For the individual sections, a table of contents is provided to show the sequence of manual pages. A notable exception to the alphabetical sequence of manual pages is the first entry at the start of each section. As a representative example, `intro.1` appears at the start of Section 1. These `intro.`*section-number* manual pages are brought to the front of each section because they introduce the

other man pages in the same section, rather than describe a command or similar provision of A/UX.

Each of the reference manuals includes at least one complete section of man pages. For example, the *A/UX Command Reference* contains sections 1 and 6. However, since Section 1 (User Commands) is so large, this manual is divided into two volumes, the first containing Section 1 commands that begin with letters A through L, and the second containing Section 6 commands and Section 1 commands that begin with letters M through Z. The sections included in each volume are as follows.

A/UX Command Reference contains sections 1 and 6. Note that both of these sections describe commands and programs available to the general user.

- Section 1—User Commands
  The commands in Section 1 may also belong to a special category. Where applicable, these categories are indicated by the letter designation that follows the section number. For example, the N in `ypcat`(1N) indicates networking as described following.

  1C Communications commands, such as `cu` and `tip`.

  1G Graphics commands, such as `graph` and `tplot`.

  1N Networking commands, such as those which help support various networking subsystems, including the Network File System (NFS), Remote Process Control (RPC), and Internet subsystem.

- Section 6—User Commands
  This section contains all the games, such as `cribbage` and `worms`.

A/UX Programmer's Reference contains sections 2 through 5.

- Section 2—System Calls
  This section describes the services provided by the A/UX system kernel, including the C language interface. It includes two special categories. Where applicable, these categories are indicated by the letter designation that follows the section number. For example, the N in `connect`(2N) indicates networking as described following.

  2N Networking system calls

  2P POSIX system calls

- Section 3—Subroutines
  This section describes the available subroutines. The binary versions are in the system libraries in the `/lib` and `/usr/lib` directories. The section includes six special categories. Where applicable, these categories are indicated by the letter designation that follows the section number. For example, the N in `mount`(3N) indicates networking as described following.

  3C C and assembler library routines

  3F Fortran library routines

  3M Mathematical library routines

  3N Networking routines

  2P POSIX routines

  3S Standard I/O library routines

  3X Miscellaneous routines

- Section 4—File Formats
  This section describes the structure of some files, but does not include files that are used by only one command (such as the assembler's intermediate files). The C language `struct` declarations corresponding to these formats are in the `/usr/include` and `/usr/include/sys` directories. There is one special category in this section. Where applicable, these categories are indicated by the letter designation that follows the section number. For example, the N in

`protocols`(4N) indicates networking as described following.

4N Networking formats

- Section 5—Miscellaneous facilities
  This section contains various character sets, macro packages, and other miscellaneous formats. There are two special categories in this section. Where applicable, these categories are indicated by the letter designation that follows the section number. For example, the P in `tcp`(1P) indicates a protocol as described following. by the letter designation in parenthesis at the top of the page:

  5F Protocol families

  5P Protocol descriptions

A/UX System Administrator's Reference contains sections 1M, 7 and 8.

- Section 1M—System Maintenance Commands
  This section contains system maintenance programs such as `fsck` and `mkfs`.
- Section 7—Drivers and Interfaces for Devices
  This section discusses the drivers and interfaces through which devices are normally accessed. While access to one or more disk devices is fairly transparent when you are working with files, the provision of *device files* permits you more explicit modes with which to access particular disks or disk partitions, as well as other types of devices such as tape drives and modems. For example, a tape device may be accessed in automatic-rewind mode through one or more of the device file names in the `/dev/rmt` directory (see `tc`(7)). The **FILES** sections of these manual pages identify all the device files supplied with the system as well as those that are automatically generated by certain A/UX configuration utilities. The names of the man pages generally refer to device names or device driver names, rather than the names of the device files themselves.
- Section 8—A/UX Startup Shell Commands
  This section describes the commands that are available from within the A/UX Startup Shell, including detailed descriptions of

those that contribute to the boot process and those that help with the maintenance of file systems.

4. How a manual entry is organized

The name for a manual page entry normally appears twice, once in each upper corner of a page. Like dictionary guide words, these names appear at the top of every physical page. After each name is the section number and, if applicable, a category letter enclosed in parenthesis, such as (1) or (2N).

Some entries describe several routines or commands. For example, `chown` and `chgrp` share a page with the name `chown`(1) at the upper corners. If you turn to the page `chgrp`(1), you find a reference to `chown`(1). (These cross-reference pages are only included in *A/UX Command Reference* and *A/UX System Administrator's Reference*.)

All of the entries have a common format, and may include any of the following parts:

NAME
is the name or names and a brief description.

SYNOPSIS
describes the syntax for using the command or routine.

DESCRIPTION
discusses what the program does.

FLAG OPTIONS
discusses the flag options.

EXAMPLES
gives an example or examples of usage.

RETURN VALUE
describes the value returned by a function.

ERRORS
describes the possible error conditions.

FILES
lists the filenames that are used by the program.

SEE ALSO
provides pointers to related information.

DIAGNOSTICS
discusses the diagnostic messages that may be produced. Self-explanatory messages are not listed.

WARNINGS
points out potential pitfalls.

BUGS
gives known bugs and sometimes deficiencies. Occasionally, it describes the suggested fix.

5. Locating information in the reference manuals

The directory for the reference manuals, *A/UX Reference Summary and Index*, can help you locate information through its index and summaries. The tables of contents within each of the reference manuals can be used also.

5.1 Table of contents

Each reference manual contains an overall table of contents and individual section contents. The general table of contents lists the overall contents of each volume. The more detailed section contents lists the manual pages contained in each section and a brief description of their function. For the most part, entries appear in alphabetic order within each section.

5.2 Commands by function

This summary classifies the A/UX user and administration commands by the general, or most important function they perform. The complete descriptions of these commands are found in *A/UX Command Reference* and *A/UX System Administrator's Reference*. Each is mentioned just once in this listing.

The summary gives you a broader view of the commands that are available and the context in which they are most often used.

5.3 Command synopses

This section is a compact collection of syntax descriptions for all the commands in *A/UX Command Reference* and *A/UX System Administrator's Reference*. It may help you find the syntax of commands more quickly when the syntax is all you need.

5.4 Index

The index lists key terms associated with A/UX subroutines and commands. These key terms allow you to locate an entry when you don't know the command or subroutine name.

The key terms were constructed by examining the meaning and usage of the A/UX manual pages. It is designed to be more discriminating and easier to use than the traditional permuted index, which lists nearly all words found in the manual page NAME sections.

Most manual pages are indexed under more than one entry; for example, `lorder`(1) is included under "archive files," "sorting," and "cross-references." This way you are more likely to find the reference you are looking for on the first try.

5.5 Online documentation

Besides the paper documentation in the reference manuals, A/UX provides several ways to search and read the contents of each reference from your A/UX system.

To see a manual page displayed on your screen, enter the `man`(1) command followed by the name of the entry you want to see. For example,

```
man passwd
```

To see the description phrase from the NAME section of any manual page, enter the `whatis` command followed by the name of the entry you want to see. For example,

```
whatis apropos
```

To see a list of all manual pages whose descriptions contain a given keyword or string, enter the `apropos` command followed by the word or string. For example,

```
apropos remove
```

These online documentation commands are described more fully in the manual pages `man`(1), `whatis`(1), and `apropos`(1) in *A/UX Command Reference*.

Table of Contents

Section 1M: System Maintenance Commands

NAME

`intro` — introduction to system maintenance commands

DESCRIPTION

This section describes, in alphabetical order, commands that are used chiefly for system maintenance and administration purposes. The commands in this section should be used along with those listed in Section 1 of the *A/UX Command Reference*. Cross references in the form *name*(1M), *name*(7), or *name*(8) refer to entries in this manual. Cross references to entries in sections other than 1M, 7 and 8 refer to entries in one of the other A/UX reference manuals (see the Preface).

REPAIRING DISKS

A/UX Local System Administration includes a detailed description of using `fsck` to repair file systems. It is always a good idea to make a complete backup of a corrupt file system that contains valuable data, just in case the attempted repairs result in unnecessary data losses. Choose a backup utility that can copy data from an unmounted file system onto the backup media.

To recover from a system crash, refer to *Local System Administration.*

NAME

accept — allow lp requests

SYNOPSIS

/usr/lib/accept *destinations*

DESCRIPTION

accept allows lp(1) to accept requests for the named *destinations* (also see reject(1M)).

destination can be a printer or a class of printers. To see the status of *destinations*, use the lpstat(1) command.

FILES

/usr/lib/accept
/usr/spool/lp/*

SEE ALSO

enable(1), lp(1), lpstat(1), lpadmin(1M), lpsched(1M), reject(1M).

NAME

acctdisk, acctdusg, accton, acctwtmp — overview of accounting commands

SYNOPSIS

/usr/lib/acct/acctdisk

/usr/lib/acct/acctdusg [–p *file*] [–u *file*]

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

/usr/lib/acct/acctwtmp *reason*

DESCRIPTION

Overview

Accounting software is a set of tools (both C programs and shell procedures) that build accounting systems. acctsh(1M) describes the shell procedures built on top of the C programs.

Connect-time accounting is handled by programs writing records into /etc/utmp, (described in utmp(4)). acctcon(1M) describes programs converting /etc/utmp into session and charging records, which acctmerg(1M) then summarizes.

The A/UX system kernel performs process accounting. When a process terminates, one record per process is written to a file (normally /usr/adm/pacct). The programs in acctprc(1M) summarize this data for charging; acctcms(1M) summarizes command use. You can examine current process data with acctcom(1).

acctmerg merges and summarizes process accounting and connect time accounting (or any accounting records in the format described in acct(4)) into total accounting records (see tacct format in acct(4)). prtacct (see acctsh(1M)) formats accounting records.

acctdisk

acctdisk reads lines containing 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

acctdusg reads its standard input (usually from find –print) and computes consumption of disk resource (including indirect blocks) by login.

–p *file*
: use a password file other than /etc/passwd (see

diskusg(1M)).

-ufile
place records acctdusg doesn't charge anyone for in *file*. This is potential way to find users trying to avoid disk charges.

accton
Typing accton turns process accounting off.

file append process accounting records to this existing file (see acct(2) and acct(4)).

acctwtmp
acctwtmp writes a utmp(4) record containing the time and a reason to its standard output. The record written will be of type ACCOUNTING (see utmp(4)).

reason
a string of up to 11 characters, numbers, $, or spaces. For example, the following are suggestions for reboot and shutdown procedures, respectively:

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

FILES

/usr/lib/acct/acctdisk	
/usr/lib/acct/acctdusg	
/usr/lib/acct/accton	
/usr/lib/acct/acctwtmp	
/etc/passwd	used to convert login name to user ID
/usr/lib/acct	holds accounting commands listed in this section of the manual
/usr/adm/pacct	current process accounting file
/etc/wtmp	login/logoff history file

SEE ALSO
acctcom(1), acctcms(1M), acctcon(1M), acctmerg(1M), acctprc(1M), acctsh(1M), diskusg(1M), fwtmp(1M), runacct(1M), acct(2), acct(4), utmp(4).

NAME

acctcms — command summary from per-process accounting records

SYNOPSIS

/usr/lib/acct/acctcms [-a [-o] [-p]] [-c] [-j] [-n] [-s] [-t] *file...*

DESCRIPTION

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

-a Print output in ASCII rather than in the internal summary format. The output includes command name, number of times executed, total kcore-minutes, total CPU minutes, total real minutes, mean size (in K), mean CPU minutes per invocation, "hog factor," characters transferred, and blocks read and written, as in acctcom(1). Normally, output is sorted by total kcore-minutes.

You can use the following options only with the -a flag option.

-p Output a prime-time-only command summary.

-o Output a nonprime (offshift) time-only command summary.

Using -p and -o together produces a combination prime and non-prime-time report. The output summaries are total usage, except number of times executed, CPU minutes, and real minutes; these are split into prime and nonprime.

-c Sort by total CPU time, rather than total kcore-minutes.

-j Combine commands invoked only once under "***other".

-n Sort by number of command invocations.

-s File names encountered hereafter are in internal summary format.

-t Process all records as total accounting records. The default internal summary format splits each field into prime and nonprime-time parts. This option combines the prime and

nonprime-time parts into a single field that totals both, and is compatible with System V style `acctcms` internal summary format records.

EXAMPLE

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

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

FILES

```
/usr/lib/acct/acctcms
/usr/lib/acct/holidays
```

SEE ALSO

`acctcom`(1), `acct`(1M), `acctcon`(1M), `acctmerg`(1M), `acctprc`(1M), `acctsh`(1M), `fwtmp`(1M), `runacct`(1M), `acct`(2), `acct`(4), `utmp`(4).

BUGS

You get unpredictable output if you use `-t` on new style internal summary format files, or if you don't use it with old style internal summary format files.

NAME

acctcom — search and format process accounting files

SYNOPSIS

acctcom [-a] [-b] [-C *sec*] [-e *time*] [-E *time*] [-f] [-g *group*] [-h] [-H *factor*] [-i] [-I *chars*] [-k] [-l *line*] [-m] [-n *pattern*] [-o *ofile*] [-O *sec*] [-q] [-r] [-s *time*] [-S *time*] [-t] [-u *user*] [-v] [*file*] ...

DESCRIPTION

acctcom reads *file*, the standard input, or /usr/adm/pacct, in the form described by acct(4) and writes selected records to the standard output. Each record represents the execution of one process. The output shows:

```
COMMAND NAME
USER
TTYNAME
START TIME
END TIME
REAL (SEC)
CPU (SEC)
MEAN SIZE (K)
```

and optionally,

```
F STAT
HOG FACTOR
KCORE MIN
CPU FACTOR
CHARS TRNSFD
BLOCKS READ
```

where

F is the *fork/exec* flag: 1 for fork without exec, STAT is the system exit status, and BLOCKS READ is the total blocks read and written.

The command name has a # inserted in front of it if it was executed with superuser privileges. If a process is not associated with a known terminal, a ? is placed 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, that is, 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 flag options are:

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

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

`-C` *sec* Show only processes with total CPU time, system plus user, exceeding *sec* seconds.

`-e` *time* Select processes existing at or before *time*, given in the format *hr*[: *min*[: *sec*]].

`-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*.

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

`-g` *group* Show only processes belonging to *group*. The *group* may be designated by either the group ID or group name.

`-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*).

`-H` *factor* Show only processes that exceed *factor*, where factor is the *hog factor*, as explained in option `-h`, above.

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

`-I` *chars* Show only processes transferring more characters than the cut-off number given by *chars*.

`-k` Instead of memory size, show total kcore-minutes.

`-l`*line* Show only processes belonging to terminal `/dev/`*line*.

`-m` Show mean core size (the default).

`-n` *pattern*
Show only commands matching *pattern*. *pattern* may be a regular expression as in `ed`(1), except that + means one or more occurrences.

`-o` *ofile* Copy selected process records in the input data format to *ofile*; suppress writing on standard output.

`-O` *sec* Show only processes with CPU system time exceeding *sec* seconds.

`-q` Do not produce any output records, just produce the average statistics as with the `-a` option.

`-r` Show CPU factor (*user-time*)/(*system-time* + *user-time*).

`-s` *time* Select processes existing at or after *time*, given in the format *hr*[: *min*[: *sec*]].

`-S` *time* Select processes starting at or after *time*.

`-t` Show separate system and user CPU times.

`-v` Exclude column headings from the output.

`-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 superuser privileges, or ? which designates only those processes associated with unknown user ID's.

`acctcom` reports only on processes that have terminated; use `ps`(1) for active processes.

FILES

```
/bin/acctcom
/etc/passwd
/usr/adm/pacct?
/etc/group
```

SEE ALSO

`ksh`(1), `ps`(1), `sh`(1), `su`(1), `acct`(1M), `acctcms`(1M), `acctcon`(1M), `acctmerg`(1M), `acctprc`(1M), `acctsh`(1M), `fwtmp`(1M), `runacct`(1M), `acct`(2), `acct`(4), `utmp`(4).

BUGS

If *time* exceeds the present time, then *time* is interpreted as occurring on the previous day.

NAME

acctcon1, acctcon2 — connect-time accounting

SYNOPSIS

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

/usr/lib/acct/acctcon2

DESCRIPTION

acctcon1 reads a sequence of login/logoff records from its standard input (redirected from /etc/wtmp) and converts them to a sequence of records, one per login session, giving the following ASCII output: device, user ID, login name, prime connect time (seconds), nonprime connect time (seconds), session starting time (numeric), and starting date and time. The flag options are:

-l *file* create *file* showing the following line usage summary: 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 tracks line usage, identifies bad lines, and finds software and hardware oddities. Hanging-up, terminating login(1), and terminating the login shell each generate logoff records, so the number of logoffs is often three to four times the number of sessions. See init(1M) and utmp(4).

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

-p print input only: 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.

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

EXAMPLE

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

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

FILES

```
/usr/lib/acct/acctcon1
/usr/lib/acct/acctcon2
/etc/wtmp
/usr/lib/acct/holidays
```

SEE ALSO

`acctcom`(1), `login`(1), `acct`(1M), `acctcms`(1M), `acctmerg`(1M), `acctprc`(1M), `acctsh`(1M), `fwtmp`(1M), `runacct`(1M), `init`(1M), `acct`(2), `acct`(4), `utmp`(4).

BUGS

Date changes confuse the line usage report. Use `wtmpfix` (see `fwtmp`(1M)) to correct this.

See acctcon(1M)

See acctcon(1M)

See acct(1M)

See acct(1M)

NAME

acctmerg — merge or add total accounting files

SYNOPSIS

/usr/lib/acct/acctmerg [-a] [-i] [-p] [-t] [-u] [-v] [*file*...]

DESCRIPTION

acctmerg reads its standard input and up to nine additional files, all in the tacct format (see acct(4)). 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. Flag 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

EXAMPLE

The following sequence is useful for repairing any file kept in this format:

```
acctmerg -v < file1 > file2
```

edit file2 as desired

```
acctmerg -i < file2 > file1
```

FILES

/usr/lib/acct/acctmerg

SEE ALSO

acctcom(1), acct(1M), acctcms(1M), acctcon(1M), acctprc(1M), acctsh(1M), fwtmp(1M), runacct(1M), acct(2), acct(4), utmp(4).

See acct(1M)

NAME

acctprc1, acctprc2 — process accounting

SYNOPSIS

/usr/lib/acct/acctprc1 [*ctmp*]

/usr/lib/acct/acctprc2

DESCRIPTION

acctprc1 reads input in the form described by acct(4), adds login names corresponding to user IDs, and then writes (for each process) an ASCII line giving user ID, login name, prime CPU time (tics), nonprime CPU time (tics), and mean memory size (in memory segment units).

The file *ctmp* contains a list of login sessions, in the form described in acctcon(1M), sorted by user ID and login name. This helps it distinguish among different login names that share the same user ID. If you don't supply this file, acctprc1 obtains login names from the password file.

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.

EXAMPLE

These commands are typically used as follows:

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

FILES

/usr/lib/acct/acctprc1
/usr/lib/acct/acctprc2
/etc/passwd
/usr/lib/acct/holidays

SEE ALSO

acctcom(1), acct(1M), acctcms(1M), acctcon(1M), acctmerg(1M), acctsh(1M), cron(1M), fwtmp(1M), runacct(1M), acct(2), acct(4), utmp(4).

BUGS

Although normally run commands distinguish among login names that share user ID's, some commands (for example, those run from cron(1M)) find it difficult to do this. They can be more precisely converted by faking login sessions on the console via the acctwtmp program in acct(1M).

CAVEAT

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

See acctprc(1M)

See acctprc(1M)

NAME

chargefee, ckpacct, dodisk, lastlogin, monacct, nulladm, prctmp, prdaily, prtacct, shutacct, startup, turnacct — shell procedures for accounting

SYNOPSIS

/usr/lib/acct/chargefee *login-name number*

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

/usr/lib/acct/dodisk [-o] [*file* ...]

/usr/lib/acct/lastlogin

/usr/lib/acct/monacct *number*

/usr/lib/acct/nulladm *file*

/usr/lib/acct/prctmp [*file*...]

/usr/lib/acct/prdaily [-l] [-c] [*mmdd*]

/usr/lib/acct/prtacct *file* [*heading*]

/usr/lib/acct/shutacct [*reason*]

/usr/lib/acct/startup

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

DESCRIPTION

chargefee

chargefee charges *number* units to *login-name* and writes a record to /usr/adm/fee, to merge with other accounting records during the night.

ckpacct

You should initiate ckpacct with cron(1M). It periodically checks the size of /usr/adm/pacct. If the size exceeds *blocks* (500 by default), it invokes turnacct switch. If the number of free disk blocks in the /usr file system falls below 500, ckpacct automatically uses turnacct off to stop collecting process accounting records. When the number of free blocks again rises to 500, it reactivates accounting. This feature is sensitive to how often ckpacct is executed, usually by cron.

dodisk

cron should invoke dodisk to perform disk accounting on the special files in /etc/checklist.

-o do a slower version of disk accounting by login directory.

file specifies one or more file systems to do disk accounting on. If you use *file*, disk accounting will only be done on these file systems. If you use the `-o` flag, *file* should be mount points of mounted file systems. If omitted, they should be the special file names of mountable file systems.

lastlogin

`runacct (1M)` invokes `lastlogin` to update `/usr/adm/acct/sum/loginlog`, which shows the last date each person logged in.

monacct

You should invoke `monacct` once a month or once an accounting period.

number

indicates the month or period. If you don't supply a *number*, it defaults to the current month (1–12). This default is useful if `cron`(1M) executes `monacct` on the first day of each month.

Note: Text can be substituted for the *number* option.

`monacct` creates summary files in `/usr/adm/acct/fiscal` and restarts summary files in `/usr/adm/acct/sum`.

nulladm

`nulladm` creates *file* with mode 664 and owner and group `adm`. Various accounting shell procedures call this file.

prctmp

`prctmp` prints the session record file (normally `/usr/adm/acct/nite/ctmp`) created by `acctcon1` (see `acctcon`(1M)).

prdaily

`runacct (1M)` invokes `prdaily` to format a report of the previous day's accounting data. The report is in `/usr/adm/acct/sum/rprt`*mmdd* where *mmdd* is the month and day of the report. Typing `prdaily` prints the current daily accounting reports.

mmdd

prints the specified days' accounting reports.

`-l` prints a report of exceptional usage by login ID for the specified date. `monacct` cleans up previous daily reports and make them inaccessible to `prdaily`.

`-c` prints a report of exceptional resource usage by command. You can only use this on the current day's accounting data.

prtacct
`prtacct` formats and prints any total accounting (`tacct`) file.

If the heading option is chosen, a heading is printed on the first line of each output page, after the date and before the page number. Multiple word headings must be enclosed in double quotes.

shutacct
Invoke `shutacct` during a system shutdown (usually in `/etc/shutdown`) to turn process accounting off and append a *reason* record to `/etc/wtmp`.

startup
`/etc/rc` should call `startup` to turn accounting on when the system is brought up.

turnacct
`turnacct` is an interface to `accton` (see `acct`(1M)) which turns process accounting `on` or `off`. `turnacct switch` 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 incremented by 1 for each additional `pacct` file), then turns accounting back on. `ckpacct` calls this procedure, and thus `cron` can use it to keep `pacct` to reasonable size.

FILES

`/usr/lib/acct/chargefee`	
`/usr/lib/acct/ckpacct`	
`/usr/lib/acct/dodisk`	
`/usr/lib/acct/lastlogin`	
`/usr/lib/acct/monacct`	
`/usr/lib/acct/nulladm`	
`/usr/lib/acct/prctmp`	
`/usr/lib/acct/prdaily`	
`/usr/lib/acct/prtacct`	
`/usr/lib/acct/shutacct`	
`/usr/lib/acct/startup`	
`/usr/lib/acct/turnacct`	
`/usr/adm/fee`	accumulator for fees
`/usr/adm/pacct`	current file for per process accounting

/usr/adm/pacct*	used if pacct gets large and when executing 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 this section of the manual
/usr/adm/acct/sum	summary directory, should be saved

SEE ALSO

acctcom(1), acct(1M), acctcms(1M), acctcon(1M), acctmerg(1M), acctprc(1M), cron(1M), diskusg(1M), fwtmp(1M), rc(1M), runacct(1M), shutdown(1M), acct(2), acct(4), utmp(4).

See acct(1M)

NAME

adduser — add a user account

SYNOPSIS

adduser [-r *real-name*] [-a *address*] [-x *extension*]
[-p *home-phone*] [-g *group*] [-s *shell*] [-d *dir*] [-h *home*]
[-u *lowest*] [-U *uid*] [-i] [-c] [*login-name*]...

DESCRIPTION

adduser creates an account for each *login-name*. One or more accounts may be added with a single command; command-line options apply to all names given. For each user, a password file entry is generated, and a home directory is created. If no *login-names* are provided, adduser enters interactive mode, prompting for all values that were not specified on the command line.

adduser uses the information given, or appropriate defaults, to generate an entry suitable for inclusion in /etc/passwd. If appropriate, an entry is also generated for /etc/group. For each account created, a brief report is written to the standard output. In the interactive mode, a confirmation is requested before the final changes to /etc/passwd are made.

A new home directory is created, if necessary, and startup files (.cshrc, .kshrc, .login, .logout, and .profile) are copied in to it from /skel. Directory and file permissions are set to read, write, and execute for owner, and read and execute for group (750). The information used to create each account is stored in a README file in each new home directory.

adduser does not permit new users to be added locally to a system that is receiving its password file via the Yellow Pages (YP). Accounts for users already in the YP password database can be added locally, but login name, group ID, and password fields are those given by the YP database, rather than those specified in the adduser command line.

In interactive mode, adduser prompts for a password for each new account. In batch mode, the password field is set to `*LK*`, which causes a password to be set when the account is first used.

FLAG OPTIONS

Command line options allow administrators to override default values. The following flag options are available:

-r *real-name* Specify the real name of a person, for example, "Fred Smith" to be associated with the account.

To preserve embedded spaces, such as the space between the first and last name, place quotation marks around *real-name*. Quotes should be used to protect any blanks in the name.

-a *address* Specify an office address, for example, mail stop or building number.

-x *extension* Specify an office telephone number.

-p *home-phone* Specify a home telephone number.

-g *group* Specify the initial login group in which each user is to be placed. If omitted, `adduser` creates a unique group for each user. The group name created is of the form `gp`*gid* where *gid* is the next available numeric group ID.

-s *shell* Specify the full pathname of an executable program to use as the shell for each user added. If omitted, the default is `/bin/csh`. Other common choices are `/bin/ksh`, and `/bin/sh`.

-d *dir* Specify the full pathname of the parent of the user's home directory. By default, home directories are created as `/users/`*login-name*. This option causes a directory other than `/users` to be used. The name of the home directory is that of the new account, *login-name*. This option may not be used with the `-h` option.

-h *home* Specify the full pathname of the desired home directory. The *login-name* is not to be considered, and *home* is used as the name of the home directory. This option may not be used with the `-d` option.

-u *lowest* Specify the desired lower bound for determining a numeric user ID (UID). If omitted, `adduser` uses the first available UID ≥ 200. The *lowest* UID may only be specified from the command line; interactive mode does not prompt for this value. This option may not be used in combination with `-U`.

-U *uid* Force the numeric UID to be *uid*. The UID may only be forced from the command line; interactive mode does not prompt for this value. This

option may not be used with -u.

-i Force an interactive mode, which is normally entered only if *login-name* is omitted. This option forces adduser to prompt for a real name, address, extension, home phone, group, shell, or home directory, which was not supplied on the command line.

-c Create a USEFUL COMMAND folder in the home directory of the user.

FILES

/etc/gtmp	Temporary group file
/etc/ptmp	Temporary password file
/etc/ogroup	Old group file
/etc/opasswd	Old password file
/usr/lib/skel/*	Standard startup files (.cshrc, .login, .profile, ...)
$HOME/README	Account-information file placed in each new account

SEE ALSO

csh(1), ksh(1), sh(1), vipw(1M).

A/UX Network System Administration.

See getty(1M)

NAME

appletalk — configure and view AppleTalk® network interfaces

SYNOPSIS

appletalk [-u] [-i *interface*] [-b *hardware_interface*] [-z] [-d] [-n] [-s]

DESCRIPTION

appletalk lets you configure and view AppleTalk network interfaces and the AppleTalk network. You can use appletalk at any time to view network interface parameters or to bring an AppleTalk interface up or down. The current version of A/UX® supports only a single interface at a time, as defined in appletalkrc(4).

FLAG OPTIONS

The following flag options may be used:

-u Bring online the interface specified in appletalkrc(4)). You must be superuser to use this option. (See WARNINGS.)

-i *interface* Define the AppleTalk interface to configure or view. This parameter is a string, such as localtalk0 or ethertalk0. The default is the interface defined in appletalkrc(4).

-b *hardware_interface*
Use the *hardware_interface*. This is the hardware interface to be associated with an EtherTalk interface; it is a string such as ae0. In order for this option to work, you must use it along with the -u option. This option is useful when the node has multiple Ethernet boards installed. It associates the EtherTalk interface with the specified hardware interface. The default interface is the hardware interface defined in appletalkrc(4).

-z Ignore the zone name hint saved from the previous incarnation of appletalk and assume that there is no zone name available for the node at startup. As a result, if there are multiple zones on the cable, the system displays a menu of valid zone names for the cable. The

system puts the node into the zone that you select. This option is valid only when you use the -u option to bring an EtherTalk interface online.

-d Take offline the AppleTalk interface. The default interface is specified in appletalkrc(4). You must be superuser to use this option. (See WARNINGS.)

-n Display the AppleTalk current node address.

-s If the AppleTalk interface is active, display LAP and DDP statistics and error counts.

EXAMPLES

To bring the interface localtalk0 online, enter

```
appletalk -i localtalk0 -u
```

To display statistics and error counts, enter

```
appletalk -s
```

To bring online interface ethertalk0 on the hardware interface ae1, enter

```
appletalk -i ethertalk0 -b ae1 -u
```

FILES

/etc/appletalk
/etc/appletalkrc
/dev/appletalk/ddp/socket
/dev/appletalk/lap/*/control

WARNINGS

If you bring the appletalk interface up or down while within term, appletalk won't work for the rest of that term session. Get out of term and launch it again in order to use appletalk functions in term again.

SEE ALSO

appletalkrc(4),
"Installing and Administering AppleTalk," in *A/UX Network System Administration.*

NAME

arp — address resolution display and control

SYNOPSIS

```
/etc/arp hostname
/etc/arp -a [unix] [kmem]
/etc/arp -d hostname
/etc/arp -s hostname ether-addr [temp] [pub]
/etc/arp -f filename
```

DESCRIPTION

The arp program displays and modifies the Internet-to-Ethernet address translation tables used by the address resolution protocol (see arp(5)).

FLAG OPTIONS

With no flag options specified, the program displays the current ARP entry for *hostname*. The host may be specified by name or by number, using Internet dot notation.

- `-a` Display all of the current ARP entries by reading the table from the file kmem (default /dev/kmem) based on the kernel file unix (default /unix).
- `-d` Delete an entry for the host called *hostname*. Only the superuser can use this option.
- `-s` Create an ARP entry for the host called *hostname* with the Ethernet address *ether-addr*. The Ethernet address is given as six hex bytes separated by colons. The entry will be permanent unless the word temp is given in the command. If the word pub is given, the entry will be "published"; that is, this system will act as an ARP server, responding to requests for *hostname* even though the host address is not its own.
- `-f` Cause the file *filename* to be read and multiple entries to be set in the ARP tables. Entries in the file should be of the form

  hostname ether-addr [temp] [pub]

FILES

```
/etc/arp
/dev/kmem
```

SEE ALSO

inet(3N), arp(5), ifconfig(1M).

NAME

autoconfig — build a new up-to-date kernel

SYNOPSIS

/etc/autoconfig [-V] [-v] [-I] [-a] [-k] [-D] [-i *file*] [-o *file*] [-m *directory*] [-b *directory*] [-l *linker*] [-S *file*] [-s *directory*] [-d *directory*] [-L *loadfile*] [-M *file*] [-t *timeout*]

DESCRIPTION

autoconfig is a utility that is run to add software to the operating system when new devices are added.

FLAG OPTIONS

You can use any of the following flag options:

-V Print the current version number of /etc/autoconfig.

-v Provide verbose output and give a step-by-step account of the autoconfiguration process.

-I Call device-specific initialization routines for all modules included in the new kernel.

-a Check to see if the running kernel matches the current hardware configuration. If the kernel matches, autoconfig exits and does not build a new kernel. If the kernel requires reconfiguration, autoconfig builds a kernel and calls driver-specific initialization routines (-I). By default, autoconfig builds the new kernel in the file /unix. After building a new kernel, autoconfig reboots the kernel.

-k Patch the current running kernel (currently not implemented).

-D Display information about modules configured in the kernel input file.

-i *file*
Change the default input file. The default is /etc/config.d/newunix.

-o *file*
Change the default output file. The default is /unix.

-m *directory*
Change the default directory used to search for master files. The default is /etc/master.d.

`-b` *directory*
Change the default directory used to search for driver object files. The default is `/etc/boot.d`.

`-l` *linker*
Change the default linker program, which is `ld`(1). This option is used for cross development.

`-S` *file*
Put a list of startup programs into the file *file* from the directory specified by `-s`. This file is usually specified as `/etc/startup`.

`-s` *directory*
Change the default directory used to search for startup programs. The default is `/etc/startup.d`.

`-d` *directory*
Change the default directory used to search for initialization programs. The default is `/etc/init.d`.

`-L` *loadfile*
Cause `autoconfig` not to search the slots for devices and instead to read records from the ASCII file *loadfile*. Each record has three fields in the following order: a slot number, a board ID (a number), and a version number. Autoconfiguration continues as if these devices are in the system.

`-M` *file*
Creates an `/etc/master` file for the use of `errpt`(1M).

`-t` *timeout*
Cause `autoconfig` to call `/etc/macquery` to present a Macintosh® alert box just before rebooting the system. If *timeout* is greater than 0, the alert is automatically confirmed (OK is selected) after *timeout* seconds. Otherwise, the user must select the OK button in the alert box to continue with the reboot. `-t0` is specified in `/etc/sysinitrc` for the boot process.

FILES

/etc/autoconfig
/etc/%autoconfig
/etc/config.d/newunix
/dev/kmem
/unix

SEE ALSO

ld(1), errpt(1M), module_dump(1M), newconfig(1M), newunix(1M).

Building A/UX Device Drivers.

NAME

badblk — set or update bad block information

SYNOPSIS

badblk [-r] /dev/rdsk/c?d?s? [*blkno...*]

DESCRIPTION

badblk sets or updates bad block information for disk partitions. badblk first attempts to alter a bad block by hardware sparing. In the event that hardware sparing fails and the device supports alternate bad blocking, badblk will attempt to alternate block the bad block. Hardware sparing may fail if the device does not support hardware sparing or if the device's capacity for hardware sparing has been exceeded.

If you invoke badblk without specifying block numbers, it will search the whole device for bad blocks and print the block numbers of any that are found.

FLAG OPTIONS

-r When this option is specified, badblk will not attempt to block any bad blocks, but instead will report those that are found.

blkno
One or more block numbers separated by blanks.

EXAMPLES

```
badblk /dev/rdsk/c0d0s31
```

does a full read-verify on the whole disk associated with controller zero, drive zero. Note that the raw device must be specified.

```
badblk /dev/rdsk/c0d0s0
```

does a full read-verify on partition zero (usually the root partition) of the disk associated with controller zero, drive zero.

NOTES

badblk uses very simple tests to determine whether or not a block is bad.

The alternate block map information can be accessed and modified through the dp(1M) utility.

The badblk command does not work on floppy disks.

FILES

/usr/bin/badblk

SEE ALSO

dp(1M), altblk(4).

See brc(1M)

NAME

bcopy — interactive block copy

SYNOPSIS

/etc/bcopy

DESCRIPTION

bcopy dates from a time when neither the UNIX file system nor disk drives were as reliable as they are now. bcopy copies from and to files starting at arbitrary block (512-byte) boundaries.

bcopy asks the following questions:

to:	the file or device to copy to
offset:	the starting "to" block number
from:	the file or device to copy from
offset:	the starting "from" block number
count:	the number of blocks to copy

After count is exhausted, it repeats the from question (giving you a chance to concatenate blocks at the to+offset+count location). If you press RETURN in response to from, everything starts over.

Press RETURN twice consecutively to terminate bcopy.

FILES

/etc/bcopy

SEE ALSO

cpio(1), dd(1).

See nfsd(1M)

NAME

brc, bcheckrc, macsysinitrc, rc, sysinitrc, powerfail — system initialization shell scripts

SYNOPSIS

/etc/brc

/etc/bcheckrc

/etc/macsysinitrc

/etc/powerfail

/etc/rc

/etc/sysinitrc

DESCRIPTION

init executes sysinitrc, macsysinitrc, brc, bcheckrc, and rc via entries in /etc/inittab. sysinitrc is executed before init starts up its initial level. The other scripts are executed when the system is changed out of single user mode. powerfail executes whenever a system power failure is detected.

brc loads any programmable microprocessors with their appropriate scripts.

bcheckrc performs consistency checks to prepare the system for multiuser mode. It prompts you to check the file systems with fsck(1M).

rc starts system daemons before the terminal lines are enabled for multiuser mode. In addition, it mounts file systems and activates accounting, error logging, system activity logging, and the Remote Job Entry (RJE) system.

sysinitrc performs various system initialization tasks, including setting the internal clock, checking the root file system, setting host and domain names, and running autoconfiguration.

macsysinitrc launches StartMonitor and CommandShell to provide a front end for the A/UX® boot process on the Macintosh® computer while the system initialization scripts are executed. Before doing so, macsysinitrc executes /etc/keyset, which needs to be executed before the Macintosh environment is active.

`powerfail` is invoked when the system detects a power failure. It performs any last-minute activities as desired before powering down.

FILES

```
/etc/brc
/etc/bcheckrc
/etc/rc
/etc/sethost
/etc/setmactime
/etc/sysinitrc
/etc/powerfail
```

SEE ALSO

`autoconfig`(1M), `fsck`(1M), `init`(1M), `query`(1M), `shutdown`(1M), `startup`(1M), `inittab`(4), `mtab`(4).

See acctsh(1M)

NAME

chgnod — change current A/UX system nodename

SYNOPSIS

chgnod *new-name* [*kernel-file*]

DESCRIPTION

chgnod accesses the structure defined in /usr/include/sys/utsname.h:

```
struct utsname {
        char  sysname[9];
        char  nodename[9];
        char  release[9];
        char  version[9];
};
```

chgnod changes the nodename of the currently running kernel to *new-name*. *kernel-file* is the name of the kernel that was last booted. If you don't specify a *kernel-file*, /unix is assumed. *nodename* is a null-terminated string containing the name the system is known by on a communications network.

new-name must not be longer than eight characters; longer names are truncated to eight.

chgnod only changes the nodename of the kernel in memory. The next time you reboot your system, your nodename will not reflect this change. If you want to permanently change your nodename, you must edit the configuration file name.c and remake your kernel.

EXAMPLE

```
chgnod user10 /unix.current
```

changes your nodename to user10 if /unix.current was the last kernel booted.

FILES

/etc/chgnod
/usr/include/sys/utsname.h

SEE ALSO

hostname(1), uucp(1C), uname(2).

NAME

chroot — change root directory for a command

SYNOPSIS

/etc/chroot *newroot command*

DESCRIPTION

Execute *command* relative to the new root. Change initial slashes (/) in path names to *newroot* for a command and any of its children. Also, change the initial working directory to *newroot*.

```
chroot newroot command > x
```

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

Only the superuser can use this command.

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

EXAMPLE

If you have a floppy-based A/UX system disk in /dev/dsk/c8d[01]s[07] then:

```
mkdir /t
mount /dev/dsk/c8d[01]s[07]/t
chroot /t /bin/sh
```

leaves you running programs off of the floppy. To return to your original shell, exit your shell.

FILES

/etc/chroot

SEE ALSO

chdir(2).

BUGS

Be very careful when referencing special files in the new root file system.

See acctsh(1M)

NAME

clri — clear inode

SYNOPSIS

/etc/clri [-T*file-system-type*] *file-system i-number* ...

DESCRIPTION

clri zeros (clears) the inode numbered *i-number* and increments the inode generation count. The *file-system* must be a special file name referring to a device containing a file system. After clri is executed, any blocks in the affected file show up as "missing" in an fsck(1M) of the *file-system*. This command should only be used in emergencies, and extreme care should be exercised.

The -T flag option indicates the file-system type, such as 4.2 or 5.2. If this option is not used, clri attempts to determine the file-system type.

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

The primary purpose of this command is to remove a file that for some reason appears in no directory. If it is used to "zap" an inode that does appear in a directory, care should be taken to track down the entry and remove it. Otherwise, when the inode is reallocated to some new file, the old entry will still point to that 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.

EXAMPLE

```
clri /dev/rdsk/c0d0s0 65
```

where /dev/rdsk/c0d0s0 is a legitimate file system and 65 is the inode number to be cleared.

WARNING

This command should be used with caution.

FILES

/etc/clri

SEE ALSO

fsck(1M), fsdb(1M), ncheck(1M), fstyp(3), fs(4).

BUGS

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

NAME

comsat — server for biff(1)

SYNOPSIS

/usr/etc/in.comsat

DESCRIPTION

comsat is the server process which receives reports of incoming mail and notifies users if they have requested this service.

comsat receives messages on a datagram port associated with the biff(1) service specification (see services(4N)) for one line messages of the form

user@mailbox-offset

If the *user* specified is logged in to the system and the associated terminal has the owner execute bit turned on (by a biff y), the *offset* is used as a seek offset into the appropriate *mailbox* file and the first 7 lines or 560 characters of the message are printed on the user's terminal. Lines which appear to be part of the message header other than the From, To, Date, or Subject lines are not included in the displayed message.

FILES

/usr/etc/in.comsat
/etc/utmp

SEE ALSO

biff(1), services(4N).

BUGS

The message header filtering is prone to error. The density of the information presented is near the theoretical minimum.

Users should be notified of mail that arrives on machines other than the one to which they are currently logged in.

The notification should appear in a separate window so it does not interfere with the screen.

NAME

cpset — install files in specified directories

SYNOPSIS

cpset [–o] *object directory* [*mode* [*owner* [*group*]]]

DESCRIPTION

cpset installs the object file *object* in *directory*. You can specify *mode*, *owner*, and *group* of the destination file on the command line. If you omit these data, there are two possible results:

If you are using cpset with administrative permissions (that is, your user ID is less than 100), it provides the following defaults:

mode
0755

owner
bin

group
bin

If you are not an administrator, the destination file has your default mode, owner, and group.

FLAG OPTIONS

The following flag option is interpreted by cpset:

–o Move *file* to OLD*file* in the destination directory before installing the new *object*.

EXAMPLES

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

The above examples have the same effect (assuming they are used by an administrator). They copy the file echo into /bin and give 0755, bin, bin as the mode, owner, and group, respectively.

cpset uses the file /usr/src/destinations to determine the final destination of a file. This file contains pairs of pathnames separated by spaces or tabs. The first name is the "official" destination, such as /bin/echo. The second name is the new destination. For example, if you move echo from /bin to /usr/bin, the entry in /usr/src/destinations would

be:

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

When the actual installation happens, `cpset` verifies that the "old" pathname does not exist. If a file is there, `cpset` issues a warning and continues. `/usr/src/destinations` is not distributed with the system; sites use it to track local command movement. The procedures for building the source define the "official" locations of the source.

NOTES

The environment variable `ROOT` locates the destination file (in the form `$ROOT/usr/src/destinations`). This is necessary when cross generation is being done on a production system.

FILES

`/usr/bin/cpset`

SEE ALSO

`make`(1), `install`(1M).

NAME

cron — clock daemon

SYNOPSIS

/etc/cron

DESCRIPTION

cron executes commands at specified dates and times. You can schedule commands regularly with instructions in crontab files; other users can submit their own crontab file with command crontab(1). Use at(1) for commands which execute only once. Since cron never exits, you should should only execute it once. cron is listed in the /etc/inittab file and is therefore started directly by init(1M).

cron examines crontab files and at command files only during process initialization and when a file is updated using crontab. This reduces the overhead of checking for new or changed files at regularly scheduled intervals.

FILES

/etc/cron	
/usr/lib/cron	main cron directory
/usr/lib/cron/log	accounting information
/usr/spool/cron	spool area
/usr/lib/cron/queuedefs	scheduling information

SEE ALSO

at(1), crontab(1), sh(1), init(1M).

DIAGNOSTICS

A history of cron actions is recorded in /usr/lib/cron/log.

NAME

dcopy — copy file systems for optimal access time

SYNOPSIS

/etc/dcopy [-s*X*] [-a*n*] [-d] [-v] [-f*fsize* [:*isize*]] *inputfs outputfs*

DESCRIPTION

dcopy copies file system *inputfs* to *outputfs*. *inputfs* is the existing file system; *outputfs* is a file system, appropriately sized to hold the reorganized result. For best results, *inputfs* should be a raw device and *outputfs* should be a block device. dcopy should be run on unmounted file systems (in the case of the root file system, copy to a new pack). With no arguments, 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* Supplies device information for creating an optimal organization of blocks in a file. The forms of *X* are the same as the -s flag option of fsck(1M).

-a*n* Places the files not accessed in *n* days behind the free blocks of the destination file system. If -a*n* is not specified, the value defaults to 7, that is, no movement occurs.

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

-v Reports how many files were processed, and how big the source and destination free lists are.

-f*fsize*[:*isize*]
Specifies 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 interrupt and quit signals and reports on its progress. To terminate dcopy, send a quit signal and dcopy will no longer catch interrupts or quits.

FILES

/etc/dcopy

SEE ALSO

fsck(1M), mkfs(1M), ps(1).

NAME

devnm — device name

SYNOPSIS

/etc/devnm [*mount-point*]

DESCRIPTION

devnm displays the device file that is currently being referenced by *mount-point*, which must be a full pathname. When *mount-point* is / and swapping is being done on the same disk section as the root file system, devnm displays both the block device file and the swap device file.

/etc/rc (see brc(1M)) uses this command to construct a mount-table entry for the root device.

EXAMPLE

If /dev/dsk/c0d0s2 is mounted on /usr, then entering

```
/etc/devnm /usr
```

produces

```
/dev/dsk/c0d0s2 /usr
```

FILES

/etc/devnm
/dev/dsk/*
/etc/mtab

SEE ALSO

brc(1M), gd(7).

NAME

dev_kill — remove devices files within a directory

SYNOPSIS

dev_kill *number directory* [*directory...*]

DESCRIPTION

dev_kill removes all device files from the specified directories that have the major number specified by number. It is intended to be invoked from device initialization programs run by the autoconfiguration system (see autoconfig(1M) and *Building A/UX Device Drivers*)

dev_kill silently ignores specified files that are not directories. This allows one to simply use:

```
dev_kill 4 /dev /dev/*
```

without getting thousands of errors. A certain number of devices required by the system (e.g. /dev/console, etc.) have fixed (small) major numbers and should not be arbitrarily removed, especially as part of an autoconfiguration.

FILES

/etc/dev_kill
/dev/console
/dev/...

SEE ALSO

autoconfig(1M).
Building A/UX Device Drivers.

NAME

diskformat — format a disk through a driver-dependent format operation

SYNOPSIS

diskformat [-dens *n*] [-head 0] *floppy-device*
diskformat [-cyl *s*[-[*e*]]] [-size 532] *hard-disk-device*

DESCRIPTION

diskformat initializes a hard or floppy disk by passing any specified parameters through the corresponding generic device interface, such as fd(7) or gd(7). This means that the parameters for diskformat can be interpreted differently for different classes of devices. For example, one form of this command can be used to format hard disks, and in that case the head option is ignored. However, the head argument is honored whenever it is passed through the fd (floppy) interface.

For an Apple® Hard Disk SC, the preferred method is to use Apple HD SC Setup to format and partition disks. That way, Apple HD SC Setup, as well as other Macintosh® utilities, are able to read the partition map and discover the whereabouts of any Macintosh file systems.

Before diskformat actually formats a disk, it issues the following message

```
About to format device. Type return to continue:
```

and waits for you to confirm the operation. This gives you a final opportunity to cancel the format operation, because it overwrites any previous data and programs on the media referenced as *floppy-device* or *hard-disk-device*. Note that any response at all, other than the interrupt or suspend character, causes diskformat to continue its operation.

You must specify a raw *floppy-device* or *hard-disk-device* for this command, as described in fd(7) and gd(7).

Without any options, the floppy disk is formatted at the highest recording density supported by the media and the drive, as long as *floppy-device* references one of the following autodensity device files:

```
/dev/rfloppy[01]
/dev/rfd/d[01]
```

However, if the diskformat command is specified without any options and the floppy media is referenced through a device file of fixed density, the media is formatted at the expected density, if at all possible. See fd(7) for a list of the device files with fixed densities.

FLAG OPTIONS

The following flag options may be used:

`-cyl` *s*`[-[`*e*`]]` Format cylinders starting from *s* and ending with *e*. A specification such as *s*- means starting from *s* and proceeding to the end of the media. Note that while start and end cylinders can be specified for any device, they are only honored if the device driver supports them. See Section 7 for details about particular device drivers, such as fd(7).

`-dens` *n* Indicates formatting density for floppy disks only. A value of `400` specifies 400K single-sided, `720` specifies 720K, `800` specifies 800K double-sided, and `1440` specifies 1440K.

`-head 0` Format a floppy disk for single-sided use (400K). This option is available as well as the `400` argument for `-dens` so that drives referenced as fixed density can be forced to honor the command without reporting errors.

`-size 532` Format the hard disk at 532 bytes per physical disk block for compatibility with some early Macintosh hard disk drives. Logical blocks remain 512 bytes per block. A/UX® simply ignores the extra tag bytes at the beginning of each physcial block.

FILES

`/bin/diskformat`

SEE ALSO

fd(7), gd(7).

NOTES

Before floppy disks can be used with commands such as `tar` or `cpio`, they must be formatted using `diskformat` or the Macintosh Operating System.

Writing to a floppy disk previously formatted under the Macintosh Operating System, using utilities such as `tar` and `cpio`, destroys any previously recorded Macintosh data and programs.

Before a `diskformat` operation is started, the device is accessed in exclusive-use mode. This prevents anyone from formatting the media in a drive already being used, or prevents anyone from using the device while it is formatting media.

The default formatting density chosen in the absence of the `dens` and `head` options is determined by the floppy device driver based on the *floppy-device* device file specified, the type of floppy drive, and the media inserted. See `fd`(7) for details.

NAME

diskusg — generate disk accounting data by user ID

SYNOPSIS

diskusg [-i *ignlist*] [-p *pw-file*] [-s] [-u *outfile*] [-v] [*file...*]

DESCRIPTION

diskusg generates intermediate disk accounting information from data in *file*. diskusg outputs lines on the standard output (one line per user) in the following format:

uid login #blocks

where

uid	the user's numeric user ID.
login	the user's login name; 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, *file* is the special filename of these devices.

diskusg recognizes the following options:

-i *ignlist*	Ignore the data on those file systems whose file system name is in *ignlist*. *ignlist* lists file system names separated by commas or enclosed within quotes. diskusg compares each name in this list with the file system name stored in the volume ID (see labelit(1M)).
-p *pw-file*	Use *file* as the name of the password file to generate login names. /etc/passwd is used by default.
-s	The input is already in diskusg output format. diskusg combines all lines for a single user into a single line. Input is supplied in a file or standard input if no file is specified.
-u *outfile*	Write records to *outfile* for files that are charged to no one. Records consist of the special file name, the inode number, and the user ID.

-v Verbose. Print a list on standard error of all files that are charged to no one.

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

EXAMPLES

The following generates daily disk accounting information:

```
for i in /dev/dsk/c0d0s0 /dev/dsk/c1d0s0; do
      diskusg $i > dtmp.`basename $i` &
done
wait
diskusg -s dtmp.* | sort +0n +1 | acctdisk > disktacct
```

FILES

/usr/lib/acct/diskusg
/dev/dsk/c0d0s0
/dev/dsk/c1d0s0
/etc/passwd converts user ID to login name

SEE ALSO

acct(1M), acctsh(1M), acct(4).

See acctsh(1M)

NAME

dp — perform disk partitioning

SYNOPSIS

dp [-q] [-u] *file*

DESCRIPTION

dp is used to perform disk partitioning and "Block Zero Block" manipulation on *file*. In most cases, though not required, *file* is a special file. dp accepts commands from standard input and performs the specified operations. dp could be considered a special purpose editor.

All input between the character # and a newline (inclusive), defines a comment and is ignored. This allows for commands to be easily kept in a disk file and piped to dp whenever a disk needs to be reinitialized. Commands do not have to be separated by newlines (that is, more than one command can be entered at an input prompt). White space is normally stripped, but may be escaped by a backslash (\). This is useful when including white space in string input. A leading zero (0) in numeric input indicates octal conversion and a leading 0x or 0X indicates hexadecimal conversion; otherwise, decimal conversion is used.

dp has some safeguards built in to help avoid destruction or deletion of data. On occasions when dp detects such a request, it will prompt you for confirmation before the action is performed. By specifying some commands in uppercase, the safeguards can be circumvented. Commands that fall into this category are so noted by the word (force) in their description.

dp has the following operation "modes."

```
command
DPME-field
BZB-field
timestamp-field
ABM-field
```

A field mode is entered when a request is made to add or modify one of the fields. Each field mode accepts its own commands. A menu containing the list of commands valid for the current mode can be printed using the ? command. A mode is exited using the q command.

dp will accept commands to manipulate and change partitions, to display and save changes, to manipulate the fields of a DPME (Disk Partition Map Entry) structure, to manipulate the fields of a BZB (Block Zero Block) structure, to manipulate the *timestamp* fields of a BZB structure, and to manipulate the fields of an ABM (Alternate Block Map) structure. For more information about these structures see altblk(4), bzb(4), and dpme(4).

In command mode, dp accepts the following commands.

a	add partition
A	add partition (force)
c	change partition
d	delete partition
f	change name of output file
i	initialize disk
I	initialize disk (force)
p	print a partition
P	print allocated partitions
q	quit dp
Q	quit dp (force)
s	print partition map status
U	uninitialize the map
v	print dp version information
w	write changes
?	print current menu
#	start of comment

In DPME-field mode, dp accepts the following commands.

a	change ABM (bzb_abm)
b	change/initialize BZB
n	name (dpme_dpident.dpiname)
t	type (dpme_dpident.dpitype)
[	physical start (dpme_pblock_start)
]	physical length (dpme_pblocks)
(	logical start (dpme_lblock_start)
)	logical length (dpme_lblocks)
>	writable (dpme_writable)
<	readable (dpme_readable)
p	print current DPME
q	quit DPME changes
?	print current menu
#	start of comment

In BZB-field mode, dp accepts the following commands.

- c no inode level badblk handling (bzb_crit)
- i bad block inode number (bzb_inode)
- m timestamps (bzb_tmade, bzb_tmount, bzb_tumount)
- n cluster number (bzb_cluster)
- p print current bzb
- q quit bzb changes
- r root FS (bzb_root)
- t FS type (bzb_type)
- T FS type (force) (bzb_type)
- u usr FS (bzb_usr)
- U uninitialize bzb
- ? print current menu
- # start of comment

In timestamp-field mode, dp accepts the following commands.

- p print timestamps
- c change creation time (bzb_tmade)
- m change mount time (bzb_tmount)
- u change umount time (bzb_tumount)
- q quit timestamp changes
- ? print current menu
- # start of comment

In ABM-field mode, dp accepts the following commands.

- b size of map in blocks (abm_size)
- c consistency check
- e number of used entries (discouraged) (abm_ents)
- I initialize altblk map per abm contents
- i default initialization of abm/altblk map
- o physical block of start of altblk map (abm_start)
- p print current abm
- q quit abm changes
- s size of map in bytes (discouraged) (abm_size)
- ? print current menu
- # start of comment

Similar to most ordinary editors, dp will encourage you to save any changes that might have been made before quitting.

LIMITS

dp will not allow the creation of a disk partition map containing more than 1024 entries.

FLAG OPTIONS

The following flag options are interpreted by dp:

-q When this option is specified, dp will not prompt for input.

-u This option will cause output to be unbuffered.

EXAMPLES

The following is an example of running dp noninteractively. It should be noted that if one was doing disk partitioning on many disks, it might prove useful to have a dpscript.

```
cat << EOF | dp -q /dev/rdsk/c0d0s31
#!dp
# @(#)dpscript  2.1
#
I64             # Initialize the map with 64 entries
# add some partitions
#
# FS type key:  1=UNIX   2=Autorec.   3=Swap
#      [       ]    Name          Type (    )   EC FS RFS UFS
a1     128   4096 Autorecovery\ 1  y   0   4096 0  2
a2    4224 111184 A/UX\ Root       y   0 111184 0  1   y   y
a3 115408  32768 Swap              y   0  32768 0  3
a4 152272   4096 Autorecovery\ 2  y   0   4096 0  2

wq              # Write changes and quit
EOF
```

To print a description of the currently defined partitions, do the following.

```
echo P | dp -q /dev/rdsk/c0d0s31
```

FILES

/bin/dp

SEE ALSO

pname(1M), altblk(4), bzb(4), dpme(4), ptab(4), autorecovery(8).

DIAGNOSTICS

dp produces various messages if the specified file does not exist.

WARNINGS

At various times while fields are being modified, dp may produce warning messages if it is determined that the modification made is of a questionable nature. Even though dp doesn't produce a warning, never assume that a modification of questionable nature hasn't been made. In the event that a modification of questionable nature is made while running as noninteractive (for example, redirecting standard input from a disk file), dp exits.

NAME

dslipuser — display the current state of slip lines on a slip server

SYNOPSIS

/etc/dslipuser

DESCRIPTION

dslipuser is used to display the /etc/slip.user file on the slip server. The /etc/slip.user file records the current number of slip users on the system and the number of available slip interfaces.

EXAMPLES

The command

```
/etc/dslipuser
```

displays

```
No dialup SLIP users connected
(2 free lines)
```

when all the slip interfaces are free.

Sample output from /etc/dslipuser when slip lines are active is

```
User user1 connected as user1-slip (aa.bb.cc.dd) via sl0
User user3 connected as user3-slip (ee.ff.gg.hh) via sl1
(0 lines free)
```

The host names *user1-slip* and *user3-slip* are from the file /etc/slip.hosts, which maintains the mapping between user log-in names and slip host names or addresses.

DIAGNOSTICS

If the modes for the user file /etc/slip.user are incorrect or if it does not exist, then a message is returned indicating that the program was unable to open that file.

FILES

/etc/slip.user

SEE ALSO

mkslipuser(1M), slip(1M), slip.user(4).

NAME

dump.bsd, rdump — copy the files within the named file system to a dump.bsd archive

SYNOPSIS

/etc/dump.bsd [-T*file-system-type*] [*key*]... [*argument...*] [*filesystem*]
/etc/rdump [-T*file-system-type*] [*key*]... [*argument...*] [*filesystem*]

DESCRIPTION

dump.bsd and rdump copy to the backup device any files within *filesystem* that have been changed after a certain date. rdump allows use of a remotely connected backup device (see the -f flag option). The *key* specifies the date and other options about the dump and consists of characters from the set 0123456789bcfusdFwWn.

FLAG OPTIONS

The -T flag option indicates the file-system type, such as 4.2 or 5.2. If this option is not used, dump.bsd attempts to determine the file-system type.

If more than one *key* is used that requires an associated *argument*, then the arguments must be supplied in the same order as each *key*.

The following options may be used for the value of *key :*

0-9 Set the "dump level" to the one-digit value specified. All files modified since the last date stored in the file /etc/dumpdates for the specified file system at lesser levels are dumped. If a date is not determined by the level, the beginning of time is assumed; thus the flag option 0 causes the entire file system to be dumped.

b Use the associated *argument* as the blocking-factor for the records of the backup device, rather than the default blocking-factor of 1. This option should only be used with the raw versions of device files. The letters b, k, m, or f may be used at the end of the associated argument to indicate a number of blocks, kilobytes, megabytes, or feet, respectively.

c Set the values for flag options b and s to those appropriate for the Apple® Tape Backup 40SC, including a default size of 37.5 MB and a blocking-factor of 8K. If a value is also

specified for the size of the media by using the s option, then it is interpreted as the number of disk blocks rather than the number of feet of tape.

f Place the dump on the associated *argument* instead of the default device file /dev/tape. If /etc/rdump is used, the associated *argument* should include a reference to the system where the backup device is located. A colon separates the remote-system name from the device file, as in

```
/etc/rdump -rbf 8k server:/dev/rmt/tc3
```

If the environmental shell variable TAPE is set and the f option is not used, the value of TAPE is used as the device file to which the output is written. If the f option is specified along with an associated argument of -, dump.bsd writes to standard output.

F Set the values for flag options b and s to those appropriate for dual-density, 3.5-inch disks, including a default size of 800K. If a value is also specified for the size of the media by using the s option, then it is interpreted as the number of disk blocks rather than the number of feet of tape.

u If the dump completes successfully, write the date of the beginning of the dump on the file /etc/dumpdates. This file records a separate date for each file system and each dump level. The contents of /etc/dumpdates is readable as text, consisting of one free-format record per line: file-system name, increment level, and ctime(3) format dump date. If necessary, /etc/dumpdates may be edited to change any of the fields.

s Specify the size of the backup media in feet. The number of feet is taken from the associated *argument*. The letters b, k, or m may be used at the end of the associated argument to indicate a number of blocks, kilobytes, or megabytes instead of feet. When the specified size is reached, dump.bsd waits for the next floppy disk or tape volume. The default tape size is 2300 feet.

d Specify the density of the tape, expressed in BPI and taken from the associated *argument*. This is used in calculating the amount of tape used per reel. The default is 1600.

W Cause dump.bsd to print out, for each file system in /etc/dumpdates, the most recent dump date and level,

and highlight those file systems that should be dumped. dump.bsd tells the operator what file systems need to be dumped. This information is gleaned from the files /etc/dumpdates and /etc/fstab. If the W flag option is set, all other flag options are ignored, and dump.bsd exits immediately.

w Similar to W, but print only those file systems that need to be dumped.

n Whenever dump.bsd requires operator attention, notify, by a means similar to a wall(1), all of the operators in the group operator.

If *filesystem* is a block device and is listed in /etc/fstab, dump.bsd will use the corresponding raw device instead.

If no arguments are given, *key* is assumed to be 9u, and a default file system is dumped to the default tape. The default file system is the root file system (/), and the default tape is /dev/tape.

dump.bsd requires operator intervention on these conditions: end of tape, end of dump, tape write error, and tape open or disk read errors (if there are more than a threshold of 32). In addition to alerting all operators implied by the n key, dump.bsd interacts with the operator on the dump.bsd control terminal at times when dump.bsd can no longer proceed, or if something is grossly wrong. All questions dump.bsd poses *must* be answered by typing yes or no, appropriately.

Making a full dump involves a lot of time and effort, so dump.bsd initiates a checkpoint at the start of each tape volume. If writing to that volume fails for some reason, dump.bsd, with operator permission, restarts itself from the checkpoint after the old tape is rewound and removed and a new tape has been mounted.

dump.bsd informs the operator of its progress at periodic intervals, including usually low estimates of the number of blocks to write, the number of tapes or floppy disks it will take, the time before completion, and the time remaining before the tape change. The output is verbose, so others will know that the terminal controlling dump.bsd is busy and will be busy for some time.

To perform dumps, start with a full (level 0) dump

```
dump.bsd 0un
```

Next, dumps of active file systems are taken on a daily basis by using a modified Tower of Hanoi algorithm with this sequence of dump levels:

```
3 2 5 4 7 6 9 8 9 9 ...
```

For the daily dumps, a set of 10 tapes per dumped file system is used on a cyclical basis. Each week, a level 1 dump is taken, and the daily Hanoi sequence repeats with 3. For weekly dumps, a set of 5 tapes per dumped file system is used, also on a cyclical basis. Each month, a level 0 dump is taken on a set of fresh tapes that is saved forever.

FILES

`/etc/dump.bsd`	
`/etc/dumpdates`	new format dump date record
`/etc/fstab`	dump table: file systems and frequency
`/etc/group`	to find group `operator`
`/dev/tape`	default tape unit to dump to

SEE ALSO

`cpio`(1), `tp`(1), `finc`(1M), `restore`(1M), `rdump`(1M), `tar`(1M), `volcopy`(1M), `fstyp`(2), `dump.bsd`(4), `fstab`(4).

DIAGNOSTICS

`dump.bsd` exits with zero status on success. Startup errors are indicated with an exit code of 1; abnormal termination is indicated with an exit code of 3.

BUGS

Fewer than 32 read errors on the file system are ignored. Each reel requires a new process, so parent processes for reels already written just hang around until the entire tape is written.

`dump.bsd` with the `W` or `w` flag option does not report file systems that have never been recorded in `/etc/dumpdates`, even if listed in `/etc/fstab`.

It would be convenient if `dump.bsd` knew about the dump sequence, kept track of the tapes scribbled on, told the operator which tape to mount and when, and provided more assistance for the operator running `restore`.

NAME

errdead — extract error records from a crash dump

SYNOPSIS

/etc/errdead *dumpfile* [*namelist*]

DESCRIPTION

When the system detects hardware errors, it generates an error record containing pertinent information about the error. If the error-logging daemon errdemon(1M) is not active or if the system crashes before the record is placed in the error file, the system holds the error information in a local buffer. errdead examines a system dump, extracts error records, and passes them to errpt(1M) for analysis.

The argument *dumpfile* specifies the file (or memory) to examine. You can specify the system namelist with *namelist*; /unix is the default.

FILES

/etc/errdead	
/unix	system namelist
/usr/bin/errpt	analysis program
/usr/tmp/err*XXXXXX*	temporary file

DIAGNOSTICS

Diagnostics may come from either errdead or errpt.

SEE ALSO

errdemon(1M), errpt(1M).

NAME

errdemon — error-logging daemon

SYNOPSIS

/usr/lib/errdemon [*file*]

DESCRIPTION

The error logging daemon errdemon collects error records from the operating system by reading the special file /dev/error and places them in *file*. If you don't specify *file* when activating the daemon, it uses /usr/adm/errfile. errdemon creates *file* if it does not exist; otherwise, it appends error records to *file* so that it doesn't lose previous error data. errdemon does not analyze the error records, this is done by errpt(1M). A software kill signal (see kill(1)) terminates the error-logging daemon. Only the superuser may start the daemon, and only one daemon may be active.

FILES

/usr/lib/errdemon	
/dev/error	source of error records
/usr/adm/errfile	repository for error records

SEE ALSO

kill(1), errpt(1M), errdead(1M), errstop(1M), error(7).

NAME

errpt — process a report of logged errors

SYNOPSIS

errpt [-a] [-*dev*] [-e *date*] [-f] [-p *n*] [-s *date*] [*file*...]

DESCRIPTION

errpt processes data collected by the error logging mechanism (errdemon(1M)) and generates a report of that data. The default report summarizes all errors posted in the named files. The options described below apply to all files. If you don't specify a file, errpt attempts to use /usr/adm/errfile.

A summary report lists: the options that may limit its completeness, the time stamped on the earliest and latest errors, and the total number of errors of one or more types. Each device summary contains: the total number of unrecovered errors, recovered errors, errors which couldn't be logged, I/O operations on the device, and miscellaneous activities on the device. It also includes as read errors the number of times that errpt has difficulty reading input data.

In addition to specific error information, any detailed report contains all instances of the error logging process being started and stopped, any time changes (via date(1)) that took place during the interval being processed, and an appended summary of each error type included.

A report may be limited to certain records in the following ways:

-a a detailed report including all error types.

-*dev*
: a detailed report is limited to data about device *dev*, where *dev* is a device identifier. errpt is familiar with the common form of identifiers (see Section 7 of this volume). The devices for which errors are logged are system dependent. Additional identifiers are int and mem which include detailed reports of stray-interrupt and memory-parity type errors, respectively.

-e *date*
: ignore all records posted later than *date*, where date has the form *mmddhhmmyy*.

-f a detailed report reporting only unrecovered block device errors.

`-p` *n*
limit the detailed report to *n* pages.

`-s` *date*
ignore all records posted earlier than *date*, where *date* has the form *mmddhhmmyy*.

FILES

`/usr/bin/errpt`	
`/etc/master`	for configuration of devices in system
`/usr/adm/errfile`	default error file

SEE ALSO

`date`(1), `errdead`(1M), `errdemon`(1M), `errfile`(4).

BUGS

When illegal options are specified, `errpt` ignores them and generates default ouput.

NAME

errstop — terminate the error-logging daemon

SYNOPSIS

/etc/errstop [*namelist*]

DESCRIPTION

errstop terminates the error-logging daemon errdemon(1M). It does this by executing ps(1) to determine the daemon's identity and then sending it a software kill signal (see signal(3)); it uses /unix as the system namelist if you don't specify one. Only the superuser may use errstop.

FILES

/etc/errstop
/unix

SEE ALSO

ps(1), errdemon(1M), kill(2), signal(3).

NAME

escher — autorecovery administration

SYNOPSIS

escher [-y] [-m]
escher *file...*

DESCRIPTION

escher determines if the regular files in the autorecovery file systems (see autorecovery(8)) are out of date with respect to the root or usr file systems. If a file is determined to be out of date, a message will appear asking if the new file should be copied to the autorecovery file system. escher will optionally mail to root a list of out of date files, copy newer files to the autorecovery file systems or, if given a list of filenames, will add these files to the Configuration Master List (CML) and copy these files to the autorecovery file systems. The CML is described in cml(4).

Note: escher may be run only by the superuser.

A hard disk may be divided into partitions. Each partition contains one file system (see fs(4)). Information on all the partitions on a disk is kept in the disk partition map (see dpme(4)) for that disk. One of the fields in a dpme is the cluster number. This is used to identify the group of partitions escher should use.

escher will only recognize partitions which reside on one disk. escher will read the cluster number from nvram, (see nvram(7)) and locate all the partitions in that cluster. A cluster must contain a root partition, a swap partition, and at least one autorecovery partition. A cluster may also contain a partition known as the usr partition. The usr partition contains a file system that is intended to be mounted on /usr. There may be multiple autorecovery partitions in a cluster.

An autorecovery file system contains copies of regular files that are needed for a minimal multiuser A/UX system. There are no special files in autorecovery file systems. The CML is a list of files required for multiuser A/UX and resides on the root file system. escher will check each file that is listed in the CML; escher ignores all special files and directories. If any of these files are invalid or are newer than the corresponding file in the autorecovery file systems, the user will be asked if the file should be copied to the autorecovery file systems.

For each file that is copied, a new CML entry will be made. A CML entry contains a list of "rules" about the file that specify what the file's attributes should be. There are CML rules for the following file attributes:

file type
linked filename
size
modification time
ownership
permissions
major and minor device numbers
version
checksum

The CML entry that is created by `escher` will have different values from the previous entry and will have a limited number of attribute specifications. This entry will have rules for:

filename
file type
size
modification time
ownership
permissions

`escher` flag options are:

`-m` Mail to `root` a list of files in autorecovery file systems that are no longer faithful copies and may need to be updated. With this option alone, no files will be copied to autorecovery file systems.

`-y` Assume a "yes" response to questions that would be asked about copying files to autorecovery file systems. It is advisable to first run `escher` with `-m` and verify that all of the listed files are "good" before using this option. It is possible that a file could be corrupt and should not be copied to the autorecovery file systems.

`escher` with both `-y` and `-m` will update all the files in autorecovery file systems that are necessary and mail a list of changed files to `root`. The `-y` and `-m` flag options may not be used when a list of files is present on the command line.

When `escher` is given a list of filenames, it will create an entry in the CML list for each file and copy it to the autorecovery file systems. The filenames must be full pathnames, beginning with `/`. The CML entry will have simple rules as described above. The named files should be files that are not currently in the CML and do not reside in autorecovery file systems.

FILES

`/etc/escher`
`/etc/eschatology/init2files`

SEE ALSO

`cml`(4), `dpme`(4), `fs`(4), `nvram`(7), `autorecovery`(8).

DIAGNOSTICS

The diagnostic messages are intended to be self-explanatory.

If an autorecovery file system runs out of space while `escher` is copying new files to it, `escher` will not attempt to update any of the other files contained in that file system. `escher` will display a message that indicates the file system is full.

WARNINGS

When `escher` is given a list of files to be added to the CML, the user must include the directories the files reside in. In order for a file to be restored through autorecovery, all the directories in the pathname of a file must also be in the CML list.

`escher` requires that the CML file be sorted by filename.

NAME

etheraddr — get an Ethernet address

SYNOPSIS

/etc/etheraddr [*slot*]

DESCRIPTION

etheraddr prints the Ethernet address stored in ROM on the board in slot number *slot*.

DIAGNOSTICS

etheraddr exits with the return status 0 if an Ethernet interface and valid ROM are available. A nonzero exit status indicates failure to find or read an Ethernet address for the host.

FILES

/etc/etheraddr

SEE ALSO

slots(3X), ae(5), arp(5P), inet(5F), intro(5).

NAME

eu — update autorecovery files

SYNOPSIS

/etc/eu *file*

DESCRIPTION

eu is used to maintain the files needed by the autorecovery feature of A/UX (see autorecovery(8)). It copies *file* to the autorecovery partition(s) and updates the relevant entry in /etc/eschatology/init2files. If *file* is not found in /etc/eschatology/init2files, an entry as described in cml(4) will be created.

To prevent inconsistent updates while eu is running, a lockfile, /etc/eschatology/FCML.lock, is used to single-thread the updates to the file systems and the cml(4) file. Once eu is complete, this lockfile is removed.

FILES

/etc/eu	
/etc/eschatology/init2files	the data base
/etc/eschatology/FCML.lock	the lock file

SEE ALSO

autorecovery(8), escher(1M), cml(4).

NAME

eupdate — update important files for autorecovery purposes

SYNOPSIS

/etc/eupdate

DESCRIPTION

eupdate updates appropriate system files for autorecovery use. This command should be used after a machine has been reconfigured with autoconfig(1M) or after modification of important relevant files (see below).

FILES

/etc/HOSTNAME
/etc/NETADDRS
/etc/eupdate
/etc/inittab
/etc/startup.d/BNET
/etc/startup.d/ae6
/unix

SEE ALSO

autoconfig(1M), eu(1M), autorecovery(8).

NAME

exterr — turn on/off the reporting of extended errors

SYNOPSIS

exterr /dev/*devicename* [*choice*]

DESCRIPTION

exterr turns on (or off) the reporting of extended errors on the specified device.

choice may be y or n (for "'yes" or "no") to turn error reporting on or off, respectively.

If reporting of errors is turned "off" with the argument n, only fatal errors are reported.

The default *choice* is "yes" (y), in which case soft as well as hard errors are reported on the specified device. The precise determination of what error messages are printed is specific to each device driver. In general, though, exterr may be used to reduce the amount of error information displayed. The devicename must be the "raw" one to access the ioctl.

FILES

/bin/exterr

EXAMPLE

```
exterr /dev/xxxx n
```

turns to off the reporting of extended errors for device /dev/*xxxx*.

NAME

ff — list file names and statistics for a file system

SYNOPSIS

/etc/ff [-a*n*] [-c*n*] [-i*inode-list*] [-I] [-l] [-m*n*] [-n*file*] [-p*prefix*] [-s] [-u] *special*

DESCRIPTION

ff reads the *special* file's ilist and directories, assumes that it is a file system, and saves inode data for files matching the selection criteria. ff outputs the path name for each saved inode, and any other file information you requested with the options described below. Output fields are positional. The output is sorted by inode, with the fields separated by tabs. ff's default output line is:

path-name inumber

If you enable all the options, the output fields are:

path-name inumber size uid

In the following list, *n* is a decimal integer (optionally signed), where +*n* means more than *n*, -*n* means less than *n*, and *n* means exactly *n*. A day is a 24 hour period.

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

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

-i *inode-list* — generate names for only those inodes specified in *inode-list*. (An *inode-list* is a comma-separated list of inode numbers).

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

-l — list path names for multiply linked files.

-m *n* — select if the file has been modified in *n* days.

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

-p *prefix* — add *prefix* to each generated path name. . is the default.

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

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

EXAMPLE

```
ff -I /dev/dsk/c0d0s0
```

generates a list of file names on the specified file system.

```
ff -m -l /dev/dsk/c0d0s0 > /log/incbackup/usr/tuesday
```

produces an index of files and inumbers on the file system which have been modified in the last 24 hours.

```
ff -i 451,76 /dev/rdsk/c0d0s0
```

obtains the path names for inodes 451 and 76 on the file system.

FILES

`/etc/ff`

SEE ALSO

`find`(1), `finc`(1M), `frec`(1M), `ncheck`(1M).

BUGS

Generates only a single path name for a multiply linked inode, unless you specify the `-l` option. When you specify `-l`, no selection criteria apply to the names generated. It includes all possible names for every linked file on the file system in the output.

On very large file systems, memory may run out before `ff` does.

NAME

finc — fast incremental backup

SYNOPSIS

finc [-a *n*] [-c *n*] [-m *n*] [-n *file*] *file-system raw-tape*

DESCRIPTION

finc selectively copies the input *file-system* to the output *raw-tape*. Mount the input *file-system* read-only to ensure an accurate backup, although you can obtain acceptable results in read-write mode. The tape must be previously labeled by labelit (see volcopy(1M)).

We recommend using the ff command to produce an index of the tape's contents before using finc. You can recover files on a finc tape with the frec command.

The argument *n* in the following options is 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 hours.

-a *n* true if the file has been accessed in *n* days.

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

-m *n* true if the file has been modified in *n* days.

-n *file* true for any file which has been modified more recently than the argument *file*.

EXAMPLE

```
finc -m -2 /dev/rdsk/c0d0s0 /dev/rmt/0m
```

writes a tape of all files from the /usr file-system modified in the last 48 hours.

FILES

/bin/finc

SEE ALSO

cpio(1), ff(1M), frec(1M), volcopy(1M).

NAME

fingerd — remote user information server

SYNOPSIS

/usr/etc/in.fingerd

DESCRIPTION

fingerd is a simple protocol based on RFC742 that provides an interface to the name and finger programs at several network sites. The program reports status information about either the system at the moment or a particular person in depth. There is no required format and the protocol consists mostly of specifying a single "command line".

fingerd listens for TCP requests at port 79. Once connected it reads a single command line terminated by a <CR><LF> which is passed to finger(1). fingerd closes its connections as soon as the output is finished.

If the line is null (i.e. just a <CR><LF> is sent) then finger returns a "default" report that lists all people logged into the system at that moment.

If a login name is specified (so fingerd receives eric<CR><LF>), then more extensive information is provided for that user, whether logged in or not. Allowable user names in the command line include both login names and user names. If a name is ambiguous, all possible derivations are returned.

FILES

/usr/etc/in.fingerd

SEE ALSO

finger(1).
RFC742 (DNN Network Information Center, SRI International)

BUGS

Connecting directly to the server from a TIP or an equally narrow-minded TELNET-protocol user program can result in meaningless attempts at option negotiation being sent to the server, which will foul up the command line interpretation. fingerd should be enhanced to filter out IAC's and perhaps even respond negatively (IAC WON'T) to all option commands received.

NAME

finstall — install A/UX commercial software from floppy disks

SYNOPSIS

finstall

DESCRIPTION

finstall provides a standard and consistent method for installing software from floppy disks onto an A/UX system. finstall displays a series of messages and prompts during the installation process. finstall prompts you for which floppy drive to use and for the directory in which to install the software. You can use the default answer by pressing return. You can optionally specify certain default answers for finstall by creating a .finstallrc or /etc/finstallrc file before running finstall.

finstall checks to make sure you have enough space on the installation directory to install the software. finstall also displays the list of files that will be installed before actually installing them. finstall then prompts you one last time for permission to proceed with the installation.

You can stop the finstall procedure at anytime by giving it an interrupt, which is normally the CONTROL-C key.

finstall creates, in a subdirectory of /etc/finstall.d, a list of the files that were installed; the list is placed into the *vendorname*/*swname*/*vname*/installedfiles file. For example, if you have installed the pqr software version 1.0 from the XYZ Company, finstall creates the list of files that were installed in XYZ/pqr/1.0/installedfiles in the directory /etc/finstall.d.

FILES

/usr/bin/finstall

SEE ALSO

newunix(1M), finstallrc(4).

NOTES

Users of the csh command shell will need to run the rehash command after installing new software so that the PATH variable is updated.

NAME

frec — recover files from a backup tape

SYNOPSIS

/etc/frec [–p*path*] [–f*reqfile*] *raw-tape inumber*:*name* ...

DESCRIPTION

frec recovers the files identified by *inumber* from the specified *raw-tape*. This is a backup tape written by volcopy(1M) or finc(1M). The data for each recovery request is written into the *name* file.

If any directories are missing in the paths of recovery *names* they will be created.

–f*reqfile* specify a file containing recovery requests. The format is

inumber:*name*

with one request per line.

–p*path* specify a *path* for names not beginning with / or ./. The default pathname is your current working directory.

EXAMPLE

```
frec /dev/rmt/0m 1216:junk
```

recovers a file, with inumber 1216, into the file junk in your current working directory.

```
frec -p /usr/src/cmd /dev/rmt m 14156:a 1232:b
3141:/usr/joe/a.c
```

recovers files with inumbers 14156, 1232, and 3141 into files:

```
/usr/src/cmd/a
/usr/src/cmd/b
/usr/joe/a.c
```

FILES

/etc/frec

SEE ALSO

cpio(1), ff(1M), finc(1M), volcopy(1M).

BUGS

While creating the intermediate directories contained in a pathname, frec can only recover inode fields for those directories contained on the tape and requested for recovery.

NAME

fsck — check file-system consistency and interactively repair

SYNOPSIS

/etc/fsck -T 5.2 [-y] [-n] [-m*timeout*] [-s*X*] [-S*X*] [-t*file*] [-q] [-D*options...*] [-f] [-p*passtostart*] [*svfs-filesystem...*]

/etc/fsck [-b*block-number*] [-y] [-n] [-m*timeout*] -T 4.2 [-p*passtostart*] [*ufs-filesystem...*]

DESCRIPTION

fsck audits and interactively repairs inconsistent conditions for A/UX® file systems. If *filesystem* is not specified, fsck reads a list of default file systems from the file /etc/fstab. If the file system is consistent, then only the number of files, number of blocks used, and number of blocks free are reported. If the file system is inconsistent, the operator is prompted for confirmation to proceed before each corrective action is attempted. Frequently corrective actions result in some loss of data. The amount and severity of data lost may be determined by examining various parameters such as non-zero filesize. Typically, each consistency correction is gated by the operator's yes or no response. However, if the operator does not have write permission for *filesystem*, fsck merely indicates corrective actions needed.

fsck has more consistency checks than its predecessors check, dcheck, fcheck, and icheck combined.

Checking the raw device is almost always faster and should be used with everything but the root file system. In addition, any file system other than the root file system should be unmounted at the time that it is checked with fsck. (It is not possible to unmount the root file system.)

FLAG OPTIONS

The flag options for the fsck command differ depending on the type of file system.

The following flag options apply to both System V file systems (SVFS) and Berkeley 4.2 file systems (UFS):

-T*file-system-type*
: Indicate the file-system type, for example, 4.2 for a Berkeley 4.2 file system (UFS) or 5.2 for a SVFS file system. If this option is not used, fsck attempts to determine the type.

`-y` Assume a yes response to all questions asked by `fsck`.

`-n` Assume a no response to all questions asked by `fsck`. This flag option does not open the file system for writing.

`-m`*timeout*
Use a Macintosh® user interface. This causes the `StartMonitor` to move the progress bar forward periodically during the boot sequence. In addition, if `fsck` finds a problem with a file system, it calls `/etc/macquery` to post a Macintosh alert box asking the user if he or she would like to repair the file system. If the user clicks the default Repair button in the alert box, `fsck` assumes a yes response to all further questions regarding that file system. If the user clicks Don't Repair, `fsck` assumes a no response to all further questions regarding that file system. If a timeout value greater than 0 is given, the dialog automatically selects the default button after that number of seconds. If *timeout* is not given, the default is 0, indicating that the alert should not autmatically time out.

`-p`*passtostart*
Similar to the `-q` option, but mid-progress phase messages are also suppressed. Besides quietly fixing (''preening'') certain file-system inconsistencies, the `-p` flag option and *passtostart* number provide another way to specify which file systems to check. *passtostart* specifies a threshold value that triggers the checking of a file system depending on its *passno* field in `/etc/fstab` (see `fstab`(4)). The default *passtostart* number is 1. If the value of the pass number is 2, as it normally is for the in `/etc/bcheckrc`, only those file systems in `/etc/fstab` with pass numbers of 2 or greater are checked. Only partitions in `fstab` that are mounted `rw` or `ro` are subject to being checked this way. If the superblock state field indicates that the file system was properly unmounted, it is skipped.

The following options are interpreted by `fsck` for a SVFS file system:

`-s`*X*
Ignore the actual free list and (unconditionally) reconstruct a new one by rewriting the superblock of the file system. The

file system should be unmounted during this operation, and if this is not possible, care should be taken that the system is quiescent and that it is rebooted immediately afterward. This precaution is necessary so that the obsolete main memory copy of the superblock does not continue to be used, or to be written onto the file system.

The `-s`*X* flag option allows for creating an optimal free-list organization. The argument *X* should be in the format *blocks-per-cylinder*:*blocks-to-skip*. If *X* is not given, the values used when the file system was created are used. If these values were not specified, then the value `400:7` is used.

`-S`*X*

Conditionally reconstruct the free list. This flag option is like `-s`*X* except that the free list is rebuilt only if no discrepancies were discovered in the file system. Using `-S` forces a no response to all questions asked by `fsck`. This flag option is useful for forcing free-list reorganization on uncontaminated file systems.

`-t`*file*

Use a scratch file if `fsck` cannot obtain enough memory to maintain its tables. If the `-t` option is specified, *file* is used, if needed, as the scratch file. Without the `-t` flag, `fsck` prompts the operator for the name of a scratch file. The file, when chosen, should not be on the file system being checked, and if it is not a special file or did not already exist, it is removed when `fsck` completes.

`-q` Suppress progress messages as well as eliminate the need to confim certain corrective actions. Unreferenced `fifos` are silently removed. If `fsck` detects inconsistencies, counts in the superblock are automatically fixed, and the free list salvaged. Inconsistencies other than these still require confirmation.

`-D`*options*

Check directories for bad blocks. If *options* is empty, the directories are merely checked. The `B` option checks for and clears parity bits in filenames, `C` checks whether all trailing characters in the filename are null, and `CZ` checks and writes nulls to all trailing characters in the filename.

`-f` Use a fast check to check blocks and sizes (Phase 1) and the free list (Phase 5). The free list is reconstructed (Phase 6) if necessary.

`-p`*passtostart*
Similar to the `-q` option, but midprogress phase messages are also suppressed. Besides quietly fixing (preening) certain file-system inconsistencies, the `-p` flag option and *passtostart* provide another way to specify which file systems to check. The value of *passtostart* specifies a threshold value that triggers the checking of a file system depending on its *passno* field in `/etc/fstab` (see `fstab`(4)). The default number for *passtostart* is 1. If the value of the pass number is 2, as it normally is for the in `/etc/bcheckrc`, only those file systems in `/etc/fstab` with pass numbers of 2 or greater are checked. Only SVFS partitions in `fstab` that are mounted `rw` or `ro` are subject to being checked this way. If the superblock state field indicates that the file system was properly unmounted, it is skipped.

The following flag option is unique to `fsck` for a Berkeley 4.2 (UFS) file system:

`-b`*block-number*
Use the block specified immediately after the flag as the superblock for the file system. Block 32 is always an alternate superblock.

Consistency Checks Performed

Inconsistencies checked are as follows:

1. Blocks claimed by more than one inode or the free list are checked.
2. Blocks claimed by an inode or the free list outside the range of the file system are checked.
3. Incorrect link counts are checked.
4. Size checks are performed to check for:
   incorrect number of blocks
   directory size not correctly aligned
5. Checks for bad inode format are performed.
6. Checks for blocks not accounted for anywhere are performed.
7. Directory checks are performed to find:
   files pointing to unallocated inodes
   inode numbers out of range

8. Superblock checks are performed for:
   more than the maximum number of inodes.
   more blocks for inodes than there are in the file system
9. Checks for a bad free-block list format are performed.
10. Checks for incorrect total free block or free inode count, or both, are performed.

Reconnection of dislocated files

Orphaned files and directories (allocated but unreferenced) are, with the operator's confirmation, reconnected, if not empty, by placing them in the `lost+found` directory. The user is notified if the file or directory is empty or not. If it is empty, `fsck` silently removes them and forces the reconnection of nonempty directories. The name assigned is the inode number. The only restriction is that the directory `lost+found` must preexist in the root of the file system being checked and must have empty slots in which entries can be made. This is accomplished by making a `lost+found` directory, copying a number of files to the directory, and then removing the files before `fsck` is executed. See `mklost+found`(1M).

EXAMPLES

```
fsck /dev/rdsk0/c0d0s0
```

checks the consistency of the file system referred through `/dev/rdsk0/c0d0s0`.

FILES

`/etc/fsck`
`/etc/ufs/fsck`
`/etc/svfs/fsck`
`/etc/%fsck`

`/etc/fstab`	Contains default list of file systems to check

SEE ALSO

`clri`(1M), `fsirand`(1M), `mkfs`(1M), `ncheck`(1M), `newfs`(1M), `typefs`(1M), `fstab`(4), `fs`(4).

"Checking the A/UX File System: `fsck`," in *A/UX Local System Administration.*

DIAGNOSTICS

The diagnostics produced by `fsck` are intended to be self-explanatory.

BUGS

Inode numbers for . and .. in each directory should be checked for validity.

NAME

fsdb — debug the file system

SYNOPSIS

/etc/fsdb [-T4.2] [-?] [-o] [-p*string*] [-w] *special*

/etc/fsdb [-T5.2] *special* [-]

DESCRIPTION

fsdb can be used to patch up a damaged file system after a crash. It has conversions to translate block and inumbers into their corresponding disk addresses. Also included are mnemonic offsets to access different parts of an inode. These greatly simplify the process of correcting control-block entries or descending the file-system tree.

Since fsdb reads the disk raw, it is able to circumvent normal file-system security. Extreme caution is advised in determining its availability on the system. Suggested permissions are 600 and owned by bin.

fsdb has different formats depending on the type of file system you are debugging. fsdb can be used for either a Berkeley 4.2 file system (UFS) or a System V file system (SVFS).

fsdb contains several error-checking routines to verify inode and block addresses. These routines can be disabled, if necessary, by invoking fsdb with the -o option for a UFS file system or by using the - option for a SVFS file system. The o command works for both file systems.

fsdb reads a block at a time and therefore works 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. Note that in order to modify any portion of the disk for a UFS file system, fsdb must be invoked with the -w option.

Wherever possible, syntax similar to adb syntax was adopted to promote the use of fsdb through familiarity.

fsdb considers numbers in UFS as hexadecimal by default and considers numbers in SVFS as decimal by default. However, the user has control over how data is to be displayed or accepted. The base command displays or sets the input/output base. Once set, all input defaults to this base, and all output is shown in this base. The base can be overriden temporarily for input by preceding hex-

adecimal numbers with `0x`, preceding decimal numbers with `0t`, or preceding octal numbers with `0`. Hexadecimal numbers beginning with a-f or A-F must be preceded with `0x` to distinguish them from commands.

Disk addressing by `fsdb` is at the byte level. However, `fsdb` offers many commands to convert a desired inode, directory entry, block, superblock, and so on, to a byte address. Once the address is calculated, `fsdb` records the result in the current address, or *dot*.

Several global values are maintained by `fsdb`: the current base (referred to as *base*), the current address (referred to as *dot*), the current inode (referred to as *inode*), the current count (referred to as *count*), and the current type (referred to as *type*). Most commands use the preset value of *dot* in their execution. For example,

```
> 2:inode
```

first sets the value of *dot* to 2, The : (colon) alerts the start of a command, and the *inode* command sets *inode* to 2. A count is specified after a , (comma). Once set, *count* remains at this value until a new command is encountered, which then resets the value back to 1 (the default). So, if

```
> 2000,400/X
```

is typed, 400 hex longs are listed from 2000, and when completed, the value of *dot* is 2000 + 400 * sizeof (long). If a RETURN is then typed, the output routine uses the current values of *dot*, *count*, and *type* and displays 400 more hex longs. A * causes the entire block to be displayed.

End of fragment, block, and file are maintained by `fsdb`. When displaying data as fragments or blocks, an error message is displayed when the end of fragment or block is reached. When displaying data using the `db`, `ib`, `directory`, or `file` commands, an error message is displayed if the end-of-file is reached. This is mainly needed to avoid passing the end of a directory or file and getting unknown and unwanted results.

Two examples showing several commands and the use of RETURN are:

```
> 2:ino; 0:dir?d
> 2:ino; 0:db:block?d
```

These two examples are synonymous for getting to the first directory entry of the root of the file system. Once there, subsequent use of RETURN (or +, -) will advance to subsequent entries. Note that these two examples

```
> 2:inode; :ls
> :ls /
```

are also synonymous.

FLAG OPTIONS

The flag options available to `fsdb` for a UFS file system are:

`-?`	Display usage.
`-o`	Override some error-conditions.
`-p`*string*	Set prompt to string.
`-w`	Open for write.

The flag option available for a SVFS file system is:

`-`	Disable error checking routines to verify inode and block addresses.

EXPRESSIONS

UFS

The symbols recognized by `fsdb` for a UFS file system are:

RETURN
: Update the value of *dot* by the current value of *type* and display using the current value of *count*.

`#`
: Indicate numeric expressions that may be composed of +, -, *, and % operators (evaluated left to right) and may use parentheses. Once evaluated, the value of *dot* is updated.

`,` *count*
: Indicate count. The global value of *count* is updated to *count*. The value of *count* remains until a new command is run. A count specifier of * attempts to show a block of information. The default for *count* is 1.

`?`*f*
: Display in structured style with format specifier *f* (see the section "Formatted Output").

`/`*f*
: Display in unstructured style with format specifier *f* (see the section "Formatted Output").

`.`
: Indicate the value of *dot*.

`+`*e*
: Increment the value of *dot* by the expression *e*. The amount actually incremented is dependent on the size of *type*:

  dot = dot + e * sizeof (*type*)

  The default for *e* is 1.

`-`*e*
: Decrement the value of *dot* by the expression *e* (see +).

`*`*e*
: Multiply the value of *dot* by the expression *e*. Multiplication and division don't use *type*. In the above calculation of *dot*, consider the sizeof *(type)* to be 1.

`%`*e*
: Divide the value of *dot* by the expression *e* (see *).

< *name* Restore an address saved in register *name*, which must be a single letter or digit.

> *name* Save an address in register *name*, which must be a single letter or digit.

=*f* Display indicator. If *f* is a legitimate format specifier (see the section "Formatted Output"), then the value of *dot* is displayed using format specifier *f*. Otherwise, assignment is assumed (see the next item).

= [*s*] [*e*]
Indicate assignment. The address pointed to by *dot* has its contents changed to the value of the expression *e* or to the ASCII representation of the quoted (") string *s*. This may be useful for changing directory names or ASCII file information.

=+ *e* Increment assignment. The address pointed to by *dot* has its contents incremented by expression *e*.

=- *e* Decrement assignment. The address pointed to by *dot* has its contents decremented by expression *e*.

SVFS

The symbols recognized by `fsdb` for a SVFS file system are

`#`	Indicate an absolute address.
`i`	Convert from an inumber to an inode address.
`b`	Convert to a block address.
`d`	Indicate directory slot offset.
`+,-`	Address arithmetic.
`q`	Quit.
`>,<`	Save and restore an address.
`=`	Indicate numerical assignment.
`=+`	Increment assignment.
`=-`	Decrement assignment.
`="  "  "  "  "  "`	Indicate character-string assignment.
`O`	Indicate error-checking flip-flop.
`p`	Indicate general print facilities.
`f`	Indicate file print facility.
`B`	Indicate byte mode.
`W`	Indicate word mode.
`D`	Indicate double-word mode.
`!`	Escape to the 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 interrupt character. If a number follows the `p` symbol, that many entries are printed. A check is made to detect block boundary overflows because logically sequential blocks are generally not physically sequential. If a count of 0 is used, all entries to the end of the current block are printed. The print options available are:

`i`	Print as inodes.
`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 inode. If followed by a number, that block of the file is printed. (Blocks are numbered from 0.) 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 checks that the block pointers used to find the data are not 0.

Dots, tabs, and spaces may be used as function delimiters but are not necessary. A line with just a newline character increments 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 inode, allowing the user to step through a region of a file system. 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 inumber and the character representation of the entry name. Inodes are printed with labeled fields describing each element.

UFS COMMANDS

A command must be prefixed by a : (colon) character. Only enough letters of the command to uniquely distinguish it are needed. Multiple commands may be entered on one line by separating them by a space, tab, or ; (semicolon).

In order to view a potentially unmounted disk in a reasonable manner, `fsdb` offers the `cd`, `pwd`, `ls`, and `find` commands. The functionality of these commands substantially matches that of their UNIX® counterparts (see individual commands for details). The '*', '?', and '[-]' wildcard characters are available.

`base=`*b*
: Display or set base. As stated above, all input and output is governed by the current *base*. If the *=b* is left off, the current *base* is displayed. Otherwise, the current *base* is set to *b*. Note that *b* is interpreted using the old value of *base*, so to ensure correctness, use the `0`, `0t`, or `0x` prefix when changing *base*. The default for *base* is hexadecimal.

`block` Convert the value of *dot* to a block address.

`cd` *dir* Change the current directory to directory *dir*. The current values of *inode* and *dot* are also updated. If no *dir* is specified, then change directories to inode 2 ("/").

`cg` Convert the value of *dot* to a cylinder group.

`directory`
: If the current *inode* is a directory, then convert the value of *dot* to a directory slot offset in that directory so that *dot* now points to this entry.

`file` Take the value of *dot* as a relative block count from the beginning of the file. The value of *dot* is updated to the first byte of this block.

`find` *dir* [`-name` *n*] [`-inum` *i*]
: Find files by name or inumber. `find` recursively searches directory *dir* and below for filenames whose inumber matches *i* or whose name matches pattern *n*. Note that only one of the two options (`-name` or `-inum`) may be used at one time. Also, `-print` is not needed or accepted.

`fill=`*p*
: Fill an area of disk with pattern *p*. The area of disk is delimited by *dot* and *count*.

`fragment`
: Convert the value of *dot* to a fragment address. The only difference between the `fragment` command and the `block` command is the amount that is able to be

displayed.

`inode` Convert the value of *dot* to an inode address. If successful, the current value of *inode* is updated as well as the value of *dot*. As a convenient shorthand, if `:inode` appears at the beginning of the line, the value of *dot* is set to the current *inode* and that inode is displayed in inode format.

`ls [-R] [-l]` *pat1 pat2 ...*
List directories or files. If no file is specified, the current directory is assumed. Either or both of the options may be used, but if used, must be specified before the filename specifiers. Also, as stated above, wildcard characters are available, and multiple arguments may be given. The long listing shows only the inumber and the name. Use the `inode` command with '`?i`' to get more information.

`override`
Toggle the value of override. Some error conditions may be overridden if override is toggled on.

`prompt` *p*
Change the `fsdb` prompt to *p*, which must be surrounded by (" ").

`pwd` Display the current working directory.

`quit` Quit `fsdb`.

`sb` Take the value of *dot* as a cylinder group number and then convert it to the address of the superblock in that cylinder group. As a shorthand, `:sb` at the beginning of a line sets the value of *dot* to the superblock and displays it in superblock format.

`!` Escape to the shell.

UFS Inode Commands

In addition to the previous commands, several commands deal with inode fields and operate directly on the current *inode* (they still require the ':'). They may be used to display or change the particular fields more easily. The value of *dot* is only used by the '`:db`' and '`:ib`' commands. On completion of the command, the value of *dot* is changed to point to that particular field. For example,

```
> :ln=+1
```

increments the link count of the current *inode* and sets the value of *dot* to the address of the link-count field.

`at` Access time.

`bs` Block size.

`ct` Creation time.

`db` Use the current value of *dot* as a direct block index, where direct blocks number from 0-11. In order to display the block itself, you need to pipe this result into the `block` or `fragment` command. For example,

```
> 1:db:block,20/X
```

would get the contents of data block field 1 from the inode and convert it to a block address. Then 20 longs are displayed in hexadecimal (see the section "Formatted Output").

`gid` Group ID.

`ib` Use the current value of *dot* as an indirect block index where indirect blocks number from 0-2. This only gets the indirect block itself (the block containing the pointers to the actual blocks). Use the `file` command and start at block 12 to get to the actual blocks.

`ln` Link count.

`mt` Modification time.

`md` Mode.

`maj` Major device number.

`min` Minor device number.

`nm` Although listed here, this command actually operates on the directory-name field. Once poised at the desired directory entry (using the `directory` command), this command allows you to change or display the directory name. For example,

```
> 7:dir:nm="foo"
```

gets the seventh directory entry of the current *inode* and change its name to foo. Note that names cannot be made larger than the field is set up for. If an attempt is made, the string is truncated to fit and a warning message to this effect is displayed.

`sz` File size.

`uid` User ID.

SVFS Inode Commands

The following mnemonics are used for inode examination and refer to the current working inode:

`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
`gen`	Generation number

FORMATTED OUTPUT

There are two styles and many format types. The two styles are structured and unstructured. Structured output is used to display inodes, directories, superblocks and the like. Unstructured output just displays raw data. The following table shows the different ways of displaying:

`?`

`c`	Display as cylinder groups.
`i`	Display as inodes.
`d`	Display as directories.
`s`	Display as superblocks.

`/`

`b`	Display as bytes.
`c`	Display as characters.
`o O`	Display as octal shorts or longs.
`d D`	Display as decimal shorts or longs.
`x X`	Display as hexadecimal shorts or longs.

The format specifier immediately follows the '/' or '?' character. The values displayed by '/b' and all '?' formats are displayed in the current *base*. Also, *type* is appropriately updated on completion.

EXAMPLES

The following two sections list examples of the `fsdb` command. Examples in the UFS file system are listed first, followed by examples in the SVFS file system.

UFS Examples

`(2000+400%(20+20))=D`
: Display 2010 in decimal (use of `fsdb` as a calculator for complex arithmetic).

`386:ino?i`
: Display inumber 386 in an inode format. This now becomes the current *inode*.

`:ln=4`
: Change the link count for the current *inode* to 4.

`:ln=+1`
: Increment the link count by 1.

`:ct=X`
: Display the creation time as a hexadecimal long.

`:mt=t`
: Display the modification time in time format.

`0:file/c`
: Display, in ASCII, block 0 of the file associated with the current *inode*.

`2:ino,*?d`
: Display the directory entries of the first blocks for the root inode of this file system. It stops prematurely if the end-of-file is reached.

`5:dir:inode; 0:file,*/c`
: Change the current inode to that associated with the fifth directory entry (numbered from 0) of the current *inode*. The first logical block of the file is then displayed in ASCII.

`:sb`
: Display the superblock of this file system.

`1:cg?c`
: Display the cylinder-group information and summary for cylinder group 1.

`2:inode; 7:dir=3`
: Change the inumber for the seventh directory slot in the root directory to 3.

`7:dir:nm= name""` Change the name field in the directory slot to `name`.

`2:db:block,*?d` Display the third block of the current *inode* as directory entries.

`3c3:fragment,20:fill=0x20` Get fragment 3c3 and fills 20 *type* elements with 0x20.

`2050=0xffff` Set the contents of address 2050 to 0xffffffff. 0xffffffff may be truncated depending on the current *type*.

`1c92434= this" is some text" " "` Places the ASCII for the string at 1c92434.

SVFS Examples

`386i` Print inumber 386 in an inode format. This now becomes the current working inode.

`ln=4` Change the link count for the working inode to 4.

`ln=+1` Increment the link count by 1.

`fc` Print, in ASCII, block 0 of the file associated with the working inode.

`2i.fd` Print the first 32 directory entries for the root inode of this file system.

`d5i.fc` Change the current inode to that associated with the fifth directory entry (numbered from 0) found from the above command. The first logical block of the file is then printed in ASCII.

`512B.p0o` Print the superblock of this file system in octal.

`2i.a0b.d7=3` Change the inumber 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""`	Change the name field in the directory slot to the given string. Quotes are optional when used with `nm` if the first character is alphabetic.
`C2b.p0d`	Print the third block of the current inode as directory entries.

WARNINGS

Extreme caution is advised in determining the availability of `fsdb` on the system. Suggested permissions are 600 and owned by bin.

SEE ALSO

`fsck`(1M), `dir`(4), `fs`(4).

NAME

fsentry — create a file-system-table entry

SYNOPSIS

fsentry −t*type* [−o *optlist*] [−d *dumpfreq*] [−p *passno*] [−n] [−f] *file-system mount-point*

DESCRIPTION

fsentry creates entries in the file-system table, /etc/fstab. Only one entry is allowed per invocation. The syntax is similar to that of the mount command. fsentry also optionally mounts the file system for which an entry has been generated.

FLAG OPTIONS

The following command-line arguments are interpreted by fsentry:

−t *type* — Specify the type of file system. This must be nfs, 4.2 (UFS), 5.2 (SVFS), swap, or ignore.

file-system — Specify the file system to be mounted. In the case of remote (NFS) file systems, this is of the form *host:mount-point* where *host* is the name of the remote host system, and *mount-point* is the full pathname of a directory on that system. In the case of local (UFS, SVFS, and swap) file systems, the form is /dev/dsk/c*x*d0s*y* , where *x* is the SCSI ID number of the hard disk containing the file system, and *y* is the slice (usually between 0 and 31, inclusive). In the case of ignore, it may be of either form.

mount-point — Specify the full pathname of a directory on the local machine, to be used as the mount point. If this directory does not exist, fsentry creates it. This option is ignored for type swap.

Additional command-line flag options allow administrators to override default values. The following options are available:

−o *optlist* — Specify a comma-separated list of mounting options, as used by mount. For remote (NFS) file systems, the default options are rw, bg, intr. For swap, 4.2 (UFS), or 5.2 (SVFS) file systems, the default is rw. For type ignore, the default is determined based on the form used for

file-system.

-d *dumpfreq* Specify the dump frequency, used by the `dump.bsd` command. The default is 0. This option is ignored for remote (NFS) file systems and swap file systems.

-p *passno* Specify the pass number, used by `fsck`. The default is 2. This option is ignored for remote (NFS) and swap file systems.

-n Specify the "No auto" option. By default, file systems are automatically mounted after a file-system-table entry is generated and each time the system boots. This option sets the `noauto` flag in the entry. The file system must be mounted explicitly with the `mount` command. `fsentry` does not mount the file system.

-f Force creation of file-system-table entry. If `/etc/fstab` already contains an existing entry for either *file system* or *mount-point*, `fsentry` prints an error message and quits. This option forces `fsentry` to generate a file-system-table entry that may overlap an existing entry. Note that no checking is done for existing entries if the type specified is `ignore`.

FILES

`/etc/fsentry`
`/etc/fstab` The file-system table

SEE ALSO

`fsck`(1M), `dump.bsd`(1M), `mount`(1M), `fstab`(4).

NAME

fsirand — install random inode generation numbers

SYNOPSIS

fsirand [-p] [-T*file-system-type*] *special*

DESCRIPTION

fsirand installs random inode generation numbers on all the inodes on device *special* and also installs a file system ID in the superblock. This helps increase the security of file systems exported by the network file system (NFS).

fsirand must be used only on an unmounted file system that has been checked with fsck(1M). The only exception is that it can be used on the root file system in single-user mode, if the system is immediately rebooted afterward.

FLAG OPTIONS

The following flag options are interpreted by fsirand:

-p Print out the generation numbers for all the inodes, but do not change the generation numbers.

-T Indicate the file-system type, for example 4.2 or 5.2. If this option is not present, fsirand attempts to determine the file-system type.

FILES

/etc/fsirand

SEE ALSO

fsdb(1M), fs(4), inode(4).

NAME

fsstat — report file-system state

SYNOPSIS

fsstat [-T*file-system-type*] *file-system*

DESCRIPTION

fsstat reports the state of the specified *file-system*. If the system was brought down cleanly or if *file-system* was successfully repaired by fsck(1M), the state should be OK.

FLAG OPTIONS

The following flag option is interpreted by fsstat:

-T*file-system-type*
: Indicate the file-system type, such as 4.2 (UFS) or 5.2 (SVFS). If this option is not used, fsstat attempts to determine the type.

FILES

/etc/fsstat
/etc/fs/*/fsstat

SEE ALSO

fsck(1M), fstyp(3), fs(4).

NAME

ftpd — Internet File Transfer Protocol server

SYNOPSIS

/usr/etc/in.ftpd [-d] [-l] [-t*timeout*]

DESCRIPTION

ftpd is the DARPA Internet File Transfer Prototocol server process. The server uses the TCP protocol and listens at the port specified in the ftp service specification; see services(4N).

If the -d flag option is specified, debugging information is written to the standard output.

If the -l flag option is specified, each ftp session is logged on the standard error output.

The ftp server will timeout an inactive session after 15 minutes. If the -t flag option is specified, the inactivity timeout period will be set to *timeout*.

The ftp server currently supports the following ftp requests; case is not distinguished.

Request	Description
ABOR	abort previous command
ACCT	specify account (ignored)
ALLO	allocate storage (vacuously)
APPE	append to a file
CDUP	change to parent of current working directory
CWD	change working directory
DELE	delete a file
HELP	give help information
LIST	give list files in a directory (''ls -lg'')
MKD	make a directory
MODE	specify data transfer *mode*
NLST	give name list of files in directory (''ls'')
NOOP	do nothing
PASS	specify password
PASV	prepare for server-to-server transfer
PORT	specify data connection port
PWD	print the current working directory
QUIT	terminate session
RETR	retrieve a file
RMD	remove a directory

RNFR	specify rename-from file name
RNTO	specify rename-to file name
STOR	store a file
STOU	store a file with a unique name
STRU	specify data transfer *structure*
TYPE	specify data transfer *type*
USER	specify user name
XCUP	change to parent of current working directory
XCWD	change working directory
XMKD	make a directory
XPWD	print the current working directory
XRMD	remove a directory

The remaining ftp requests specified in Internet RFC recognized, but not implemented.

The ftp server will abort an active file transfer only when the ABOR command is preceded by a Telnet "Interrupt Process" (IP) signal and a Telnet "Synch" signal in the command Telnet stream, as described in Internet RFC 959.

ftpd interprets file names according to the "globbing" conventions used by csh(1). This allows users to utilize the metacharacters "*?[]{}~".

ftpd authenticates users according to three rules.

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

2) The user name must not appear in the file /etc/ftpusers.

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

In the last case, ftpd takes special measures to restrict the client's access privileges. The server performs a chroot(2) command to the home directory of the "ftp" user. In order that system security is not breached, it is recommended that the "ftp" subtree be constructed with care; the following rules are recommended.

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

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

~ftp/etc)
Make this directory owned by the superuser and unwritable by anyone. The files passwd(4) and group(4) must be present for the ls command to work properly. These files should be mode 444.

~ftp/pub)
Make this directory mode 777 and owned by "ftp". Users should then place files which are to be accessible via the anonymous account in this directory.

SEE ALSO

ftp(1N).

BUGS

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

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

NAME

fuser — identify processes using a file or file structure

SYNOPSIS

/etc/fuser [–] [–k] [–n*namelist*] [–u] *file*...

DESCRIPTION

fuser lists the process IDs of the processes using the *files* specified as arguments. For block special devices, it lists all processes using any file on that device. The process ID is followed by c, p or r if the process is using the file as its current directory, the parent of its current directory (only when in use by the system), or its root directory, respectively.

– cancel any flag options currently in force before specifying a new group of files.

–k send the SIGKILL signal to each process. Only the superuser can terminate another user's process (see kill(2)).

–n specify an alternate namelist (/unix is the default).

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

You can respecify options between groups of files. The new set of flag options replaces the old set.

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

EXAMPLE

fuser –ku /dev/dsk/c1d0s0
: allow the superuser to terminate all processes preventing disk drive one from being unmounted. List the process ID and login name of each as it is killed.

fuser –u /etc/passwd
: list process IDs and login names of processes that have the password file open.

fuser –ku /dev/dsk/c1d0s0 –u /etc/passwd
: will do both of the above examples in a single command line.

FILES

/etc/fuser	
/unix	for namelist
/dev/kmem	for system image

/dev/mem	also for system image
/dev/swap	for outswapped processes

SEE ALSO

ps(1), mount(1M), kill(2), signal(3).

BUGS

fuser cannot determine what processes are using *files* on remotely mounted file systems.

NAME

fwdload — load an application onto an intelligent peripheral

SYNOPSIS

fwdload [-a] [-v] [-f*dev*] [-n*name*] *filename*

DESCRIPTION

The utility fwdload loads a program onto an intelligent peripheral. The peripheral must have a "forwarder" configured for it (see forwarder(7)). If the -f flag option is used, the peripheral is *dev;* otherwise, standard output will be used.

The -n flag option allows you to specify a string that is a name to be used instead of *filename* for the program to download. The default for this string is *filename*. The fwd_lkup command reports this string in its name field.

The -v flag option provides diagnostics in verbose format.

The parameter *filename* is the application to download. The file containing this application must be in COFF format. Before the download, a reset is issued to the intelligent peripheral.

If the [-a] option is used, there is no reset. Once the load is complete, execution of the downloaded application will begin at the START indicated by the COFF file.

EXAMPLE

```
fwdload -f /dev/fwdicp13 at_load
```

will download the AppleTalk® driver onto the default AppleTalk peripheral in slot 13.

FILES

/etc/fwdload
/etc/startup.d/fwdicp.d/at_load
/etc/startup.d/fwdicp.d/tt_load

SEE ALSO

fwd_lkup(1M), forwarder(7);
"AppleTalk Programming Guide," in *A/UX Network Applicatons Programming.*

NAME

`fwd_lkup` — look up the application loaded onto an intelligent peripheral

SYNOPSIS

`fwd_lkup` [`-f`*dev*] [`-v`]

DESCRIPTION

`fwd_lkup` looks up the name of the application loaded onto an intelligent peripheral. The peripheral must have a "forwarder" configured for it. If the `-f` flag option is used, the peripheral is *dev*; otherwise, standard input is used.

The `-v` flag option provides diagnostics in verbose format.

EXAMPLE

```
fwd_lkup -f /dev/fwdicp13
```

will find out what application is running on the ICP card in slot 13. If the card is currently running AppleTalk®, `fwd_lkup` prints the following:

```
begin start name
0     0     at_load
7fff  0     AVAIL
7fff  0     END
```

This indicates that `at_load`, the AppleTalk load module, is loaded on the ICP, and that it is occupying all 7fff bytes of the ICP's memory.

FILES

`/usr/bin/fwd_lkup`

SEE ALSO

`fwdload`(1M), `forwarder`(7); "AppleTalk Programming Guide," in *A/UX Network Applications Programming*.

NAME

fwtmp, wtmpfix — manipulate connect accounting records

SYNOPSIS

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

DESCRIPTION

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

-ic input is in ASCII form and output is to be written in binary form.

wtmpfix examines the standard input or named files in wtmp format, makes the time/date stamps on the entries consistent, and writes to the standard output. Using – in place of *files* indicates the standard input. If you don't correct the time/date stamp, acctcon1 will fault when it encounters certain date-change records.

Each time the date is set, a pair of date change records is written to /etc/wtmp. The first record is the old date marked with old time in the line field and the flag OLD_TIME in the type field of the utmp.h structure. The second record specifies the new date and is marked with new time in the line field and the flag NEW_TIME in the type field. wtmpfix uses these records to synchronize all time stamps in the file.

In addition to correcting time/date stamps, wtmpfix ensures that the name field consists solely of alphanumeric characters or spaces. If it encounters an invalid name, it changes the login name to INVALID and writes a diagnostic to the standard error. In this way, wtmpfix reduces the chance that acctcon1 will fail when processing connect accounting records.

FILES

```
/usr/lib/acct/fwtmp
/usr/lib/acct/wtmpfix
/etc/wtmp
/usr/include/utmp.h
```

SEE ALSO

acctcom(1), ed(1), acct(1M), acctcms(1M), acctcon(1M), acctmerg(1M), acctprc(1M), acctsh(1M), runacct(1M), acct(2), acct(4), utmp(4).

NAME

apm_getty, getty — set terminal type, modes, speed, and line discipline

SYNOPSIS

/etc/getty [-h] [-t*timeout*] *line* [*speed* [*type* [*linedisc*]]]
/etc/getty -c *file*
/etc/apm_getty *getty-options*

DESCRIPTION

getty is a program that is invoked by init(1M) and is the second process in the series (init-getty-login-shell) that ultimately connects a user with the A/UX® system. getty generates a login message field for the entry it is using from /etc/gettydefs. Then getty reads the user's login name and invokes the login(1) command with the user's name as argument. While reading the name, getty attempts to adapt the system to the speed and type of terminal being used.

apm_getty provides functionality beyond normal getty for use with an Apple® Personal Modem. Before it turns over control to getty, apm_getty sends the control sequence to select auto-answer mode. To switch back to a dialout line, the line containing apm_getty in /etc/inittab should be changed to off instead of respawn. To activate these changes, use init q as described in init(1M).

The name of a tty line in /dev to which getty is to attach itself is *line*. getty uses this string as the name of a file in the /dev directory to open for reading and writing. Unless getty is invoked with the -h flag option, getty forces a hangup on the line by setting the speed to 0 before setting the speed to the default or specified speed. The -t flag option plus *timeout* in seconds, specifies that getty should exit if the open on the line succeeds and no one types anything during the specified number of seconds. The optional second argument, *speed*, is a label to a speed and tty definition in the file /etc/gettydefs. This definition tells getty at what speed the interface should initially run, what the login message should look like, what the initial tty settings are, and what speed should be tried next should the user indicate that the speed is inappropriate by typing a break character. The default *speed* is 300 baud. The optional third argument, *type*, is a character string describing to getty what type of terminal is connected to the line in question. getty understands the following types:

`none`	Default
`vt61`	DEC vt61
`vt100`	DEC vt100
`c100`	Concept 100

The default terminal is `none`, that is, any CRT or normal terminal unknown to the system. Also, for the terminal type to have any meaning, the virtual terminal handlers must be compiled into the operating system. They are available, but not compiled, in the default condition. The optional fourth argument, *linedisc*, is a character string describing which line discipline to use in communicating with the terminal. Again, the hooks for line disciplines are available in the operating system, but only one is presently available, the default line discipline `LDISC0`.

When given no optional arguments, `getty` sets the *speed* of the interface to 300 baud, specifies that raw mode is to be used (awaken on every character), that echo is to be suppressed, that either parity is allowed, that newline characters are to be converted to return-line feed, and that tab expansion be performed on the standard output. It types the login message before reading the user's name one character at a time. If a null character (or framing error) is received, it is assumed to be the result of the user pushing the "break" key. This will cause `getty` to attempt the next *speed* in the series. The series that `getty` tries is determined by what it finds in `/etc/gettydefs`.

The user's name is terminated by a newline or RETURN character. The latter results in the system being set to treat RETURN characters appropriately (see `ioctl`(2)).

The user's name is scanned to see if it contains any lowercase alphabetic characters; if not, and if the name is nonempty, the system is told to map any future uppercase characters into the corresponding lowercase characters.

In addition to the standard A/UX system erase and kill characters (DELETE and CONTROL-U), `getty` also understands `\b` as an erase. `getty` sets the standard erase character or kill character to match.

`getty` also understands the "standard" ESS protocols for erasing, killing, aborting, and terminating a line. If `getty` sees the ESS erase character, `_`, or kill character, `$`, or abort character, `&`, or the ESS line terminators, `/` or `!`, it arranges for this set of characters to be used for these functions.

Finally, login is called with the user's name as an argument. Additional arguments may be typed after the login name. These are passed to login, which places them in the environment. See login(1).

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

getty attempts to prevent the communication programs cu(1), tip(1), and uucico(1M) from interfering with its operation by creating a lock file in /usr/spool/uucp. The lock file is not created by getty until input is present, so that these programs may attempt to access *line* while getty is running.

FILES

/etc/getty
/etc/gettydefs
/etc/issue
/usr/spool/uucp/LCK..*line*

SEE ALSO

ct(1C), init(1M), login(1), ioctl(2), gettydefs(4), inittab(4), tty(7).

BUGS

Although getty understands simple single-character quoting conventions, it is not possible to quote the special control characters that getty uses to determine when the end of the line has been reached, which protocol is being used, and what the erase character is. Therefore, it is not possible to login via getty and type a #, @, /, !, _, DELETE, CONTROL-U, CONTROL-D, or & as part of your login name or argument. They are always interpreted as having their special meaning, as described earlier.

See pwck(1M)

NAME

ifconfig — configure network interface parameters

SYNOPSIS

/etc/ifconfig *interface* [*address* [*dest-address*]] [*parameter...*]

/etc/ifconfig *interface* [*protocol-family*]

DESCRIPTION

ifconfig is used to assign an address to a network interface or configure network interface parameters. ifconfig must be used at boot time to define the network address of each interface present on a machine; it may also be used at a later time to redefine an interface's address or other operating parameters. The *interface* parameter is a string of the form *name unit*, for example, ae0.

A DARPA-Internet address is either a host name present in the host name data base, hosts(4), or a DARPA Internet address expressed in the Internet standard "dot notation."

The following parameters may be set with ifconfig:

up — Mark an interface "up." This may be used to enable an interface after an ifconfig down. It happens automatically when setting the first address on an interface. If the interface was reset when previously marked down, the hardware will be reinitialized.

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

trailers — Request the use of a "trailer" link level encapsulation when sending (default). If a network interface supports *trailers*, the system will, when possible, encapsulate outgoing messages in a manner which minimizes the number of memory to memory copy operations performed by the receiver. On networks that support the Address Resolution Protocol (see arp(5P); currently, only 10 Mb/s Ethernet), this flag indi-

cates that the system should request that other systems use trailers when sending to this host. Similarly, trailer encapsulations will be sent to other hosts that have made such requests. Currently used by Internet protocols only.

`-trailers` Disable the use of a "trailer" link level encapsulation.

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

`-arp` Disable the use of the Address Resolution Protocol.

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

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

`-debug` Disable driver dependent debugging code.

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

work portion.

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

`broadcast` (Inet only) Specify the address to use to represent broadcasts to the network. The default broadcast address is the address with a host part of all 1's.

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

Only the superuser may modify the configuration of a network interface.

DIAGNOSTICS

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

FILES

`/etc/ifconfig`

SEE ALSO

`netstat`(1), `rc`(1M), `intro`(5).

NAME

inetd — Internet services daemon

SYNOPSIS

/etc/inetd [-d]

DESCRIPTION

inetd is the Internet super-server which invokes all Internet server processes as needed. Connection-oriented services are invoked each time a connection is made, by creating a process. This process is passed the connection as file descriptor 0 and an argument of the form *sourcehost.sourceport* where *sourcehost* is hex and *sourceport* is decimal.

Datagram oriented services are invoked when a datagram arrives; a process is created and passed the connection as file descriptor 0. inetd will look at the socket where datagrams arrive again only after this process completes. The paradigms for such processes are either to read off the incoming datagram and then fork and exit, or to process the arriving datagram and then time out.

inetd consults servers(4) when it is invoked, and supports whatever services are in that file.

An rpc server can be started from inetd. The only differences from the usual code are that svcudp_create should be called as

```
transp = scvudp_create(0)
```

since inet passes a socket file as descriptor 0, and svc_register should be called as:

```
svc_register(PROGNUM, VERSNUM, service, transp, 0)
```

with the final flag as 0, since the program will already have been registered by inetd. If you want to exit from the server process and return control to inet, you must explicitly exit since scv_run never returns.

The format of entries in /etc/servers for rpc services is:

```
rpc udp server-program program-number version-number
```

where *server-program* is the C code implementing the server, and *program-number* and *version-number* are the program and version numbers, respectively, of the service. The keyword udp can be replaced by tcp for tcp-based services.

If the same program handles multiple versions, the version number can be specified as a range:

```
rpc udp /usr/etc/rstatd 100001 1-2
```

FLAG OPTIONS

The following flag option is interpreted by `inetd`:

`-d` Specifies that debugging traces are to be turned on for connection-oriented (TCP) services.

FILES

`/etc/servers` list of Internet server processes

SEE ALSO

`ftpd`(1M), `rexecd`(1M), `rlogind`(1M), `remshd`(1M), `talkd`(1M), `telnetd`(1M), `tftpd`(1M), `servers`(4).

BUGS

There is no provision for selectively invoking TCP debugging packet tracing per-service.

You should reread the `/etc/servers` file on receipt of a `SIGHUP` signal. The `/etc/servers` file can have no more than 26 lines.

NAME

init, telinit — process control initialization

SYNOPSIS

/etc/init [0123456SsQqabc]

DESCRIPTION

init is a general process spawner. Its primary role is to create processes based on the line entries in /etc/inittab (see inittab(4)). This file usually instructs init to spawn terminal listeners (see getty(1M)) on each serial line available for access to the system. It can also manage the system's autonomous processes, often called daemons.

For interchangeable use, telinit is linked to init (see ln(1)). Normally, the permissions are set so that telinit and init can only be run by the superuser or a member of the group sys.

Certain single-letter arguments for init are called run levels, and others are called directives.

Run level

can be a number between one and six or the letters s or S. init places the system in the specified run level. Using s or S causes init to change the virtual system teletype, /dev/syscon, to the terminal from which the command was executed, as well as to establish the corresponding run level.

Directive

can be a, b, c, Q, or q. The Q or q directive causes init to re-examine the /etc/inittab file and honor any changes that apply to the current run level. For example, a getty process that is changed from respawn to off in /etc/inittab will be killed when init q is entered. The a, b, or c directive causes init to create or remove only those processes with the corresponding letter in their *run level* field in /etc/inittab. Affected processes may be switched on or off as specified in the *action* field (see inittab(4)).

A run level can be viewed as a software configuration that must, at minimum, consist of the processes specified in /etc/inittab for that run level. init can be in one of eight run levels, 0 through 6, S or s.

Besides using init to change the run level, it can be used to help switch various processes on and off in tandem (see *directives*, previously described).

An init process is run as part of A/UX initialization, and that copy of the program must run continuously. It maintains the system state. By invoking the init program again, at the command line, the original copy of init can be instructed to change the system's run level. The newly requested copy of init sends the appropriate signals to the one created at system startup.

Following is a description of the role init plays in the startup process, and gives examples of the special processing init performs when there is a change in run level or in the status of a process (termination).

First init looks for an entry in /etc/inittab in which initdefault appears in the *action* field (see inittab(4)). If there is one, init uses the *run level* specified in that entry as the initial run level to enter. If this entry is not in inittab or inittab is not found, init requests that the user enter a run level from the virtual system console, /dev/syscon. If an S (s) is entered, init goes into the single-user level. This is the only run level that doesn't require the existence of a properly formatted inittab file. If /etc/inittab doesn't exist, then, by default, the only legal run level that init can enter is the single-user level. In the single-user level, the virtual console terminal /dev/syscon is opened for reading and writing and the command /bin/su is invoked immediately. To exit from the single-user run level, one of two flag options can be elected. First, if the shell is terminated (via an end-of-file), init reprompts for a new run level. Second, the init command can signal init and force it to change the run level of the system.

When attempting to boot the system, failure of init to prompt for a new run level may be due to the fact that the device /dev/syscon is linked to a device other than the physical system teletype (/dev/systty). If this occurs, init can be forced to relink /dev/syscon by typing a delete on the system teletype which is co-located with the processor.

When init prompts for the new run level, the operator may enter only one of the digits 0 through 6 or the letters S or s. If S is entered init operates as previously described in single-user mode with the additional result that /dev/syscon is linked to the

user's terminal line, thus making it the virtual system console. A message is generated on the physical console, /dev/systty, saying where the virtual terminal has been relocated.

When init comes up initially and whenever it switches out of single-user state to normal run states, it sets the ioctl(2) states of the virtual console, /dev/syscon, to those modes saved in the file /etc/ioctl.syscon. This file is written by init whenever single-user mode is entered. If this file does not exist when init wants to read it, a warning is printed and default settings are assumed.

If a 0 through 6 is entered, init enters the corresponding run level. Any other input will be rejected and the user will be reprompted. If this is the first time init has entered a run level other than single-user, init first scans inittab for special entries of the type boot and bootwait. These entries are performed, provided that the run level entered matches that of the entry before any normal processing of inittab takes place. In this way, any special initialization of the operating system, such as mounting file systems, can take place before users are allowed onto the system. The inittab file is scanned to find all entries that are to be processed for that run level.

Run level 2 is usually defined by the user to contain all of the terminal processes and daemons that are spawned in the multiuser environment.

In a multiuser environment, the inittab file is usually set up so that init will create a process for each terminal on the system.

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

To spawn each process in the inittab file, init reads each entry and forks a child process for each entry that should be respawned. After it has spawned all of the processes specified by the inittab file, init waits for one of its descendant processes to die, for a power-failure signal, or until it is signaled by init to change the system's run level. When one of these conditions occurs, init re-examines the inittab file. New entries can be

added to the inittab file at any time; however, init still waits for one of these conditions. To process the altered or new entries in /etc/inittab immediately, use the command init Q or init q to awaken init and make it re-examine the inittab file.

If init receives a power-failure signal (SIGPWR) and is not in single-user mode, it scans inittab for special powerfail entries. These entries are invoked (if the run levels permit) before any further processing takes place. In this way, init can perform various cleanup and recording functions whenever the operating system experiences a power failure.

When init is requested to change run levels, init sends the warning signal (SIGTERM) to all processes that are undefined in the target run level. init waits 20 seconds before forcibly terminating these processes via the kill signal (SIGKILL). When the q or Q flag option is specified init removes processes that are currently running but which are set to be off in /etc/inittab for the current run level.

FILES

/etc/init
/etc/init
/etc/inittab
/etc/utmp
/etc/wtmp
/etc/ioctl.syscon
/dev/syscon
/dev/systty

SEE ALSO

getty(1M), login(1), sh(1), who(1), kill(2), inittab(4), ioctl.syscon(4), utmp(4).
"System Startup and Shutdown" in *A/UX Local System Administration.*

DIAGNOSTICS

If init finds that it is continuously respawning an entry from /etc/inittab more than 10 times in 2 minutes, it assumes that there is an error in the command string and will generate an error message on the system console. init refuses to respawn this entry until either 5 minutes has elapsed or it receives a signal from a user init. This prevents init from monopolizing system resources when someone makes a typographical error in init-

tab file or a program is removed that is referenced in inittab.

NAME

install — install files in specified directories

SYNOPSIS

/etc/install [-c *dira*] [-f *dirb*] [-g *group*] [-i] [-m *mode*] [-n *dirc*] [-o] [-s] [-u *user*] *file* [*dirx*...]

DESCRIPTION

install is a command most commonly used in "makefiles" (see make(1)) to install *file* as an 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 flag options or directories (*dirx*...) are given, install searches 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 then it 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 are searched before the directories specified in the default list.

FLAG OPTIONS

The following flag options are interpreted by install:

-c *dira* Install a new command (*file*) in the directory specified by *dira*, only if a file with the same name is not found. If it is found, install issues a message saying that *file* already exists, and exits without overwriting it. This flag option may be used alone or with the -s flag option.

-f *dirb* Force *file* to be installed in given directory, whether or not a file by the same name already exists. If the file already exists, the mode and ownership of the file is that of the already existing file. This flag option may be used alone or with the -o or -s flag option.

-g*Ogroup*

Use the specified *group* ID instead of the default, bin, when setting the ownership of files that do not already

exist.

`-i` Ignore the default directory list, searching only through the given directories (*dirx...*). This flag option may be used alone or with any other flag options other than `-c` and `-f`.

`-m`*Omode*
Use the specified *mode* instead of the default, `775`, when setting the mode of files that do not already exist.

`-n` *dirc* Put *file* in the directory specified in *dirc* if *file* is not found in any of the searched directories. This flag option may be used alone or with any other flag options other than `-c` and `-f`.

`-o` If *file* is found, save the "found" file by copying it to `OLD`*file* in the directory in which it was found. This flag option is useful when installing a normally text-busy file such as `/bin/sh` or `/etc/getty`, where the existing file cannot be removed. This flag option may be used alone or with any other flag options other than `-c`.

`-s` Suppress the printing of messages other than error messages. This flag option may be used alone or with any other flag options.

`-u` *user* Use the specified *user* ID instead of the default, `bin`, when setting the ownership of files that do not already exist.

FILES

`/etc/install`

SEE ALSO

`cpset`(1M), `make`(1), `chown`(1), `chgrp`(1), `chmod`(1).

NAME

kconfig — tune kernel parameters for work-load optimization

SYNOPSIS

/etc/kconfig [-a [-v] [-V]] [-n*namelist*]

DESCRIPTION

kconfig manipulates an A/UX® kernel code file for changing system parameters.

> *Note:* It is not recommended that this utility be used unless you know exactly what you are doing. Incorrect use can cause system failures.

kconfig can be used to either list or change the system parameters listed later in the section "System Parameters."

Note that kconfig does not change parameters of the running kernel, just the image on the disk. You must then run shutdown(1M) and reboot(1M) for the changes to be effective.

FLAG OPTIONS

kconfig interprets the following flag options:

-a	List the current values of the parameters in the kernel object file *namelist*.
-v	Use with the -a flag option to produce verbose (commented) output.
-V	Use with the -a flag option to print the current version of the kernel object file *namelist*.
-n *namelist*	Specify which kernel object file is being modified (the default is /unix).

If -a is used, kconfig displays the parameters and exits. If -a is not used, standard input is read for a list of changes. You can specify one change per input line of the form

PARAM = *value*
PARAM += *value*
PARAM -= *value*

where *PARAM* is one of the parameter names (listed later in the section "System Parameters") and the *value* is either a decimal constant or a hexadecimal constant preceded by 0x.

If the = form is used, the parameter is given the value specified. If the -= form is used, the parameter is decreased by the value given. If the += form is used, the parameter is increased by the value. If the value is not within a system-defined maximum and minimum range for the parameter, an error occurs and the kernel is not changed.

SYSTEM PARAMETERS

The following system parameters are recognized by `kconfig`:

Note: Not all of these parameters will necessarily be supported in future releases of the operating system.

`NBUF` Specify the number of disk I/O buffers to allocate. These form a data cache for information read or written to file systems. Each buffer consists of `SBUFSIZE` data areas and about three dozen bytes of header information. Increasing the number of buffers improves the "cache hit ratio" on the buffer pool, but at the expense of available memory for processes. The number of system buffers normally ranges from 100 to 1500. One hundred buffers should be used for systems with 2 megabytes of total memory. Probably not more than 750 buffers should be used on systems with 4 MB of memory. Increasing the number of buffers reduces the memory available for applications and may cause more paging to occur. Systems with a single user might conceivably use more buffers than systems that typically have numerous memory-intensive applications running. The default value of `NBUF` is 100. If desired, the system dynamically calculates the number of buffers. If `NBUF` is set to 0, 10 percent of free memory at boot time is allocated to I/O buffers.

`SBUFSIZE` Determine the size of system I/O logical block size on disk devices. The size is configurable. Most System V file systems (SVFS) use a 1 kilobyte (KB) block size. The `SBUFSIZE` parameter determines the

size of in-core buffers allocated for the buffer cache. The number must be an even multiple of 512 bytes and should be large enough to accept the largest block size of all active file systems. When the system buffer size is increased, the total number of buffers should be decreased, assuming the same amount of memory is used for the buffer cache.

NPBUF — Specify how many physical input/output buffer headers to allocate. One header is needed for each simultaneous read or write of a "character-special" disk or tape device, or for each concurrent swap I/O. The default value is 20.

NFILE — Determines the size of the system file-table pool. Each entry represents an open file in use by some process. When no space is available in the file table, the message "`file: table is full`" is printed on the system console. The size is generally between 100 (the default) and 400. `NFILE` is often equal to the `NINODE` parameter.

NINODE — Set the size of the system inode table. Each table entry represents an in-core inode being used for an open file, an open working directory, or a mount point. For systems using a network file system (NFS) to access remote file systems, only locally open files consume inodes. Normally the `NINODE` parameter is greater than or equal to the `NFILE` parameter. Generally the range is from 100 (the default) to 400 inodes. When all inodes in the system inode table are used, the message "`inode: table is full`" is printed.

NSPTMAP — Allocate table entries that are used to map the system page table entries. The default value is 75. If the message "`sptreserve: No kernel virtual space`" is printed, the system has exhausted the map.

NCALL
: Specify the size of the timeout table. Each entry may be used by device-driver software to arrange for a function to be called at a later time. The default value is 50. If many add-on drives, such as for NuBus™ peripherals, are added to the system, this value might require an increase. If the timeout table is exhausted, the message "`timeout table overflow`" is printed, and the system halts execution.

NMOUNT
: Specify the size of the SVFS mount table. The parameter does not affect the number of remotely mounted network file systems allowed. If numerous disk devices containing many file-system partitions are present, the mount table may need to be increased. The default size is 10.

NFLOCK
: Specify the number of system-wide `locking`(2) file locks. Each area of a locked file requires one of these table entries. If the table is exhausted, the error `EDEADLOCK` is returned to the application that made the lock request. The default number of locks is 200.

NREGION
: Defines the number of memory regions available to all processes in the system. A typical process has a memory region for data, a memory region for stack, and a memory region for program text, but this region would be shared between all processes executing the application. If the region table is exhausted, the message "`Region table overflow`" is printed on the system console. The default value is 200.

NPROC
: Specify the total number of processes in the system. In general, each executing command, application, or system daemon is a process. Each user of the system, or each active layer or window generally uses between 2 and 8 simultaneous processes. When no processes are available, the mes-

sage "proc: table is full" is printed at the system console. By default, NPROC is 50.

NCLIST — Specify the number of system command lists (CLIST). A CLIST is a memory area used by driver software for terminals, built-in modems, or serial printer connections. Five to 10 CLISTs are required by each active terminal. When no CLISTs are available, incoming characters are lost. Each CLIST requires 64 bytes of data, and a 12-byte header. The console and some optional serial cards use the streams interface and do not require CLISTs. The default number of CLISTs is 200.

MAXUP — Determines the maximum number of concurrent processes for each user ID. The superuser is exempt from this restriction. This limit is based on user ID, not on the login terminal. If ten people are logged in using a single user ID, the limit could be reached quickly. Normally NPROC is at least 10 percent larger than MAXUP. The default value of MAXUP is currently 25.

VHNDFRAC — The virtual memory system depends on the activity of the paging daemon vhand to free memory by paging unused memory to the swap-disk device. The algorithm writes out to disk pages that have not been used for some time. If the system is not active, there is plenty of free memory and no work for vhand. VHNDFRAC and other tuning parameters allow the adventurous system administrator to fine-tune the performance.

The vhand fraction is used to determine the initial value of the system variable VHANDL. If free memory falls below VHANDL, the paging daemon, vhand, is awakened to begin aging and monitoring the resident set of virtual memory pages. At system startup time, VHANDL is set to

vhandl=MAX(maxmem/VHANDFRAC,GETPGSHI)

where maxmem is the available free memory at startup time and GETPGSHI is the free-memory high-water mark, described later. Normally VHANDFRAC is set to 16.

MAXPMEM — Specify the maximum physical memory to use. If this is set at 0, the system uses all available physical memory. It is recommended this value be 0, unless you are testing alternate memory configurations.

NMBUFS — Allocate a number of buffers for networking. Each buffer requires 256 bytes, of which 240 are available for data. As few as 100 mbufs may be used for basic networking. When NFS is used on a system, the number should be increased. As a guide, each NFS daemon may transfer 8 KB of data. Allocating (8192*n_daemons)/240 provides a starting point in the calculation. The command netstat -m may be used to determine the number of mbufs in use. If the message "m_expand returning 0" is seen frequently or if the system halts after displaying the message "panic: out of mbufs", the number of mbufs should be increased. By default, 500 mbufs are allocated.

NPTY — Determines the total number of possible pseudo tty devices (that is, /dev/pty*). This default number, which is also the maximum potential number of devices, is 16. Special files must still be created in the /dev directory for ptys to be used. If more than the allocated number of ptys are created, the error code "No such device or address" is returned by the open(2) of any of the unallocated device files.

MAXCORE — Set the space available for use by the memory allocation of the kernel from its internal heap. Most data structures used to

access remote files via NFS are allocated from this pool, as is space used by generic disk devices and the system's name-lookup code. If the message "panic: kmem_alloc" appears and the system halts, the value of MAXCORE should be increased. The default value is 128 KB.

MAXHEADER — Limit the number of allocations possible from the kernel's internal heap. If the message "panic: getfreehdr" is seen, this allocation should be increased. The default value MAXHEADER is 2048.

NSTREAM — Determine the maximum number of stream heads possible in the system. The console, AppleTalk®, and the shell layering by the console are examples of streams. The number of streams required might range from 8 to 48. The default number of stream allocated is 32.

NQUEUE — Define the maximum number of stream queues. Each stream head, driver, and module pushed onto a stream creates two queues. Typically this parameter is set to 8 times the number of stream heads. The default number of queues is 256.

The allocation of stream blocks determines the availability of buffer space used by stream devices. The optimal allocation depends on the types of devices present in the system. It is expected that the installation scripts for devices using the streams mechanism will include the necessary commands to increase the number of blocks allocated. If too few blocks of the size required by a driver are present, the system may lose input characters.

NBLK4096 — Allocate a number of 4 KB stream blocks. The default is 0.

NBLK2048 — Allocate a number of 2 KB stream blocks. The default is 20.

NBLK1024 — Allocate a number of 1 KB stream blocks. The default is 12.

NBLK512	Allocate a number of 512-byte stream blocks. The default is 8.
NBLK256	Allocate a number of 256-byte stream blocks. The default is 16.
NBLK128	Allocate a number of 128-byte stream blocks. The default is 64.
NBLK64	Allocate a number of 64-byte stream blocks. The default is 256.
NBLK16	Allocate a number of 16-byte stream blocks. The default is 128.
NBLK4	Allocate a number of 4-byte stream blocks. The default is 512. Keyboard and tty input uses this resource.
SLICE	Specify the maximum time slice available to a process before it is considered for rescheduling. At the end of the time slice, the active process is suspended, and the system searches for a process with a higher CPU priority. If no higher-priority process exists, the previous process is given another slice. The default time slice is 60. The units are "ticks," and there are 60 ticks in a second.
GETPGSLOW	Specifies the get-pages low limit which is the free-memory low-water mark for the vhand daemon. When this number is reached, vhand becomes active and starts stealing pages from active processes. The default value is 20. Increase the value to make the daemon more active. The value must be greater than 0 and less than GETPGSHI.
GETPGSHI	Specify the get-pages high limit which is the free-memory high-water mark for the vhand daemon. When this number is exceeded, the system stops stealing pages from active processes. The default value is set to 30. GETPGSHI should be greater than GETPGSLOW and less than about one-fourth of the total available memory.

GETPGSMSK — Specify, when used by vhand which pages to steal. The default value of GETPGSMSK (the get-pages mask) is 0x408. It may be modified, but most changes are more educational than useful. Values in this parameter correspond to bits in the page-table entries. Each masked bit in the page-table entry must be 0 in order for the page to be taken by vhand. The current setting is

```
PG_NDREF | PG_REF
```

(defined in /usr/include/sys/page.h). By including the modified bit (PG_M), vhand would not steal any pages with the modified bit on.

VHANDR — Specify in seconds the maximum rate at which vhand may run if free memory is less than VHANDL, as explained earlier for VHNDFRAC. The default value is 5. Increase the value to make the daemon less active. The value should be between 1 and 10.

MAXSC — Specify the maximum number of pages that may be written to the swap device in a single operation. The default value is 64. Increasing this number increases the I/O overhead spent in swapping, but decreasing the value may reduce the amount of free memory available when a page fault occurs.

MAXFC — Specify the maximum number of pages that may be placed on the free list at one time. The default value is 100. Increasing the number may allow for faster handling of page faults when a process needs more memory, but it may also reduce the working set of applications so that page faults occur more frequently.

MAXUMEM — Determine the maximum user virtual address space in pages. This number may range from about 30 to 0x20000. The default value is 0x40000, which eliminates all checking. Small values of MAXUMEM may make software that is normally taken for granted unusable.

FLCKREC — Specify the number of `flock`(2) lock structures. When this size is exceeded, the error `ENOSPC` is returned to the requesting program. The default value is 200.

FLCKFIL — Specify the number of `flock` inode structures. When this size is exceeded, the error `EMFILE` is returned to the requesting program. The default value is 50.

CDLIMIT — Set the process `ulimit` on file size. Only this number of 512-byte blocks may be written to any file by any process owned by any user. The default value is 16 million blocks (0x1000000).

CMASK — Determine the system wide default file-creation mask. Generally, the value of `CMASK` is overridden by the `umask` directive of one's chosen shell. The default value is 0.

ROOTDEV — Determine the disk device containing the root file system. This must be a device number as used internally by the kernel. The device major number is in the upper byte, and the minor number is in the lower byte. If the value is 0xffff, then the value passed from the A/UX StartupShell (see `StartupShell`(8)) booter is used.

SWAPDEV — Specify the swap disk device. The specification is the same format as `ROOTDEV`. If set to 0xffff, the value passed from the A/UX StartupShell booter is used.

PIPEDEV — Determine the disk device for temporary pipe-file space. The specification is the same format as `ROOTDEV`. If set to 0xffff, the value passed from the A/UX StartupShell booter is used.

DUMPDEV — Currently unused by the system.

SWAPLO — Specify the starting disk address of the swap area to determines the number of 512-byte blocks to skip at the beginning of the swap partition. This would be done if these blocks were to be used for some other purpose; however, this result may also be achieved by repartitioning the

disk. The default value is 0.

SWAPCNT — Specify the size of the swap area, which is a number of 512-byte blocks. The system warns you if it is running short of swap space. If SWAPCNT is 0, the size of the swap area is set to the size of the swap partition of the root device. To adjust swap space, both kconfig and the dp(1M) (disk partitioning) utility must be used. The default value is 0.

MINARMEM — Set the minimum number of pages of physical ("resident") memory reserved for user text and data segments in order to avoid deadlock. The default value is 10, and values might range from 10 to 40.

MINASMEM — Define the minimum number of pages of system (swap + resident) memory reserved for system purposes and therefore unavailable for the text and data segments of user processes. The default value is 10. Normally MINASMEM is greater than MINARMEM.

FILES

```
/etc/kconfig
/unix
```

SEE ALSO

dp(1M), reboot(1M), shutdown(1M), swap(1M), flock(2), locking(2), open(2), pty(7), StartupShell(8).

Building A/UX Device Drivers, *A/UX Local System Administration*, *A/UX Network System Administration*, *A/UX Network Applications Programming*.

NAME

keyset — set console keyboard mapping

SYNOPSIS

/etc/keyset [*keyboard*] [*country*]

DESCRIPTION

keyset sets the current console keyboard mapping to the type of keyboard given by *keyboard* and *country*. Currently, three different keyboard types are supported by the console driver. They are ADB extended, ADB ISO, and ADB standard. They are given to keyset as one of the following arguments: adbext, adbiso, or adbstd. If this parameter is not specified, keyset reads the file /dev/kmem to find out what type of keyboard was in use when the system was booted. The different *country* types currently supported are Australia, Britian, Denmark, Dutch, Finland, Flemish, FrCanada, France, Germany, Greece1, Greece2, Iceland, Italy, Norway, Spain, Sweden, SwissFrench, SwissGerman, Turkey, USA, and Yugoslavia. If this parameter is not specified, keyset reads the default information stored in the Macintosh system file /usr/lib/mac/System for the *country* parameter.

EXAMPLES

To set the keyboard to ADB standard in the United States, use the command: /etc/keyset adbstd USA

FILES

/usr/lib/mac/System

DIAGNOSTICS

The exit status is 0 if everything went OK. A status of 1 indicates a usage error or an error while setting the new keyboard map.

NAME

killall — kill all active processes

SYNOPSIS

/etc/killall [-n *namelist*] [*signal*]

DESCRIPTION

killall is a procedure used by /etc/shutdown to kill all active processes not directly related to the shutdown procedure.

If you use the -n *namelist* option, the argument will be taken as the name of an alternate *namelist* file in place of /unix.

killall is used chiefly to terminate all processes with open files so that the mounted file systems will be unbusied and can be unmounted.

killall sends *signal* (see kill(1)) to all remaining processes not belonging to the above group of exclusions. If no *signal* is specified, a default of 9 is used.

FILES

/etc/killall
/etc/shutdown

SEE ALSO

fuser(1M), kill(1), ps(1), shutdown(1M), signal(3).

See volcopy(1M)

See acctsh(1M)

NAME

line_sane — push streams line disciplines

SYNOPSIS

/etc/line_sane [*fildes*]

DESCRIPTION

line_sane pushes the streams line discipline "line" onto the stream referenced by the open file descriptor *fildes* (an integer). If *fildes* does not reference a stream, or it references a stream that already has a line discipline pushed onto it, no line discipline is pushed. If the *fildes* argument is not specified, the default is 0. In addition, if output processing is not currently being done on *fildes*, the following flags are turned on:

c_iflag	BRKINT, IGNPAR, ISTRIP, IXON, IXANY, ICRNL.
c_oflag	OPOST, ONLCR, TAB3.
c_lflag	ISIG, ICANON, ECHO, ECHOK.

FILES

/etc/line_sane

SEE ALSO

line_push(3), console(7), streams(7), termio(7).

NAME

lockd — process network lock daemon

SYNOPSIS

/etc/rpc.lockd [-t *timeout*] [-g *graceperiod*]

DESCRIPTION

lockd is a network lock daemon that processes lock requests that are sent either locally by the kernel or remotely by another lock daemon. lockd forwards lock requests for remote data to the lock daemon at the server site through the RPC/XDR package. lockd then requests the status monitor daemon, statd(1M), for monitor service. The reply to the lock request is not sent to the kernel until the status monitor daemon and the lock daemon at the server site replies.

If either the status monitor daemon or the lock daemon at the server site is unavailable, the reply to a lock request for remote data is delayed until all daemons become available.

When a server recovers, it waits for a grace period for each lockd at a client site to submit reclaim requests. Each lockd at a client site, on the other hand, is notified by the statd of the server recovery and promptly resubmits a previously granted lock request. If lockd fails to secure a previously granted lock at the server site, lockd sends SIGTERM to a process.

FLAG OPTIONS

The following flag options are interpreted by lockd:

-t *timeout* Use *timeout* (seconds) as the interval instead of the default value (15 seconds) to retransmit a lock request to the remote server.

-g *graceperiod*
Use *graceperiod* (seconds) as the grace period duration instead of the default value (45 seconds).

SEE ALSO

fcntl(2), lockf(3), signal(3), statd(1M).

NAME

Login — present a Macintosh® login dialog box when called by init

SYNOPSIS

Login [-- [-r] [-g]]

DESCRIPTION

Login presents a Macintosh dialog box allowing a user to log into the system at the Macintosh display and optionally change his or her password. Additionally, it allows you to select which type of session to log in to (for example, A/UX Finder or Console Emulator). Login replaces the use of /etc/getty and /bin/login for the console terminal.

Login is invoked instead of getty for the console terminal by loginrc which is invoked by init after the system has booted into multiuser mode and after the previous user has logged out. If Login fails to execute for any reason, either loginrc or Login invokes getty to prevent init from endlessly respawning Login.

Login reinitializes the Macintosh virtual machine state. Since this can only be done when the Macintosh environment is not already running, Login cannot be entered as a command from within a CommandShell window.

Login runs as a standalone application rather than in a MultiFinder environment so it can properly clean up its environment when it executes the default shell of the user. For security reasons, no desk accessories are available while Login is running.

The main login dialog box displayed by Login contains two radio buttons labeled Guest and Registered User, with the latter selected initially. The Guest button is enabled only if there is an entry in the /etc/passwd file for a user named Guest. Normally, two edit fields (also called text boxes) labelled Name and Password are shown. If the Guest radio button is enabled and is selected, the word Guest is automatically entered into the name field, unless the -g field is passed to Login. The -g flag specifies that the user does not need to enter the Guest password to log in to that account and hides the name and password fields.

The Login button at the bottom of the dialog box is initially disabled. This allows the user to enter his or her name and then press the RETURN key the get the password. (Pressing the TAB key or

clicking in the password field also works.) Once the user has entered his or her name and moved the text edit cursor to the password field, the Login button is enabled. Pressing the RETURN key again after entering the password is equivalent to clicking the Login button and continues the login procedure.

When the user's name and password are verified, `Login` updates accounting files and switches to the user's home directory, user ID, and group ID, as specified in the `/etc/passwd` file entry for the user.

If the user selects the Login button, the password-aging information (if any) in the password field of the user's entry in `/etc/passwd` is examined (see `passwd`(4)) for more information on password aging). If the password has expired, or if the user has no password but one is required, a dialog box is presented asking the user to enter a new password. Once the user enters a valid new password and selects the OK button, another dialog box is presented asking the user to retype the new password for verification. After typing the new password again, the user selects the Login button in this second dialog box to continue the login process.

If the user selects the Set Password button in the main login dialog box rather than the Login button, the password-aging information is examined to ensure that the password is changeable by the user and that at least the minimum number of weeks have passed since the password was last modified. Dialog boxes are presented asking the user to enter his or her new password, as described above.

`Login` then hides the main dialog box, changes the ownership and modes of `/dev/console` and `/dev/uinter0` (the Macintosh user interface driver) so that only that user can execute Macintosh applications while he or she is logged in, and changes the ownership and modes of the Macintosh hierarchical file system (HFS) disk partitions so that they show up on the user's desktop in the Finder™.

Finally, `Login` does an `exec`(2) of the user's command interpreter or shell, which is specified in the `/etc/passwd` file entry for the user. To indicate that this invocation of the command interpreter is the login shell, the name of the interpreter is prefixed with a minus sign, for example, `-sh`. If the command interpreter field in the password file is empty, then the default command interpreter, the Bourne shell (`/bin/sh`), is used.

If the user's shell is a standard shell (as listed in /etc/shells) Login may pass the -c flag to the shell with an initial command that runs a Macintosh type session as specified in the Options menu of Login. Session types include A/UX Finder (32-bit), A/UX Finder (24-bit), and Console Emulator. For the Console Emulator session type, the -c flag option is not required. See the discussion of session types below for more information.

The basic environment (see environ(5)) is initialized to:

HOME=*your-login-directory*
PATH=*:/bin:/usr/bin:/usr/ucb:/mac/bin*
LOGNAME=*your-login-name*
TERM=mac2
SHELL=*last-field-of-passwd-entry*
MAIL=*/usr/mail/your-login-name*
FINDER_EDITOR=/mac/bin/TextEditor
EDITOR=/usr/bin/vi
DISPLAY=*hostname:0.0* (used by the X Window System)

The FINDER_EDITOR variable specifies which editor is invoked when the user double-clicks on a text document icon. EDITOR is used by many A/UX utilities.

Menu Options

In addition to the main dialog box, Login displays menus titled Options and Special in addition to the Apple menu at the far left of the menu bar. The following sections describe the action performed by the various menu items.

The Apple Menu

About Login

Display a dialog box that gives version information.

The Options Menu

Change Password

Presents a dialog box with fields asking for the user's name, old password, and new password. The name and old password fields contain the information that the user had already typed into the name and password fields of the main dialog box. The cursor is positioned at the first blank field. When the user has entered the appropriate information and presses the OK button, the password aging information is examined to ensure that the password is changeable by the user and that at least the minimum number of weeks have passed since the password was last modified. If modification of the password is permitted, this dialog box is then closed and a second dia-

log box is presented asking the user to type his or her password for verification. After pressing the OK button in this second dialog box, the name and password fields on the main dialog are updated to reflect the user's changes.

Change Session Type
Presents a dialog box allowing the user to change his session type. This dialog box also has fields asking the user for his or her name and password. These fields are initialized to contain the same information as the user has already typed into the main dialog box, if any. After pressing the OK button in the dialog box, the name and password fields in the the main dialog box are updated to reflect the user's changes.

The Special Menu
This menu is somewhat analogous to the Finder's Special menu. It contains menu items to shut down and restart the system.

Restart
Present a dialog box asking for the root password (for security purposes). If other users are logged in or other systems are mounting NFS partions from this system, it also asks for a message to send to those users and a time period to delay before restarting. After the user has responded to the dialog box and the delay period has ended, the system is restarted.

Shut Down
Shuts down A/UX. It is similar to the Restart menu item, except that when the delay period has ended, it turns the machine off.

FLAG OPTIONS
`Login` is normally invoked only by `loginrc`(1M) which is invoked by `init`(1M), as specified in `/etc/inittab` and `/etc/loginrc`. Only the superuser should have permission to modify `/etc/inittab`. Thus, only the superuser can change the behavior of `Login` by changing the command-line arguments that are passed to it.

`Login` uses some of the same startup code as `startmac`(1M). Thus, `Login` accepts the `startmac` command-line arguments, which are processed by `getopt(3C)`. See `startmac`(1M) for a description of these arguments.

Login also takes additional command line arguments. To specify these, use the getopt special option -- (two hyphens) to delimit the end of the startmac options. The additional Login options must follow this special option. The additional options are:

-r Remove the normal UNIX® System V password restrictions. See passwd(1) for a list of the restrictions.

-g Allow users to log in to the Guest account (if it exists) without entering the password. Passing this flag also causes the name and password fields to be hidden when the user chooses the Guest radio button.

Session Types

Login supports a range of session types or environments which the user may choose. The standard session types shipped with A/UX are A/UX Finder (32-bit), A/UX Finder (24-bit), and Console Emulator. A/UX Finder (32-bit) is the default, but the user can change session type as described under the Options menu.

The /mac/lib/sessiontypes directory contains files which specify session type information. There is one file per session type. Each file is a Macintosh resources file containing one string list whose ID is 128. The string list contains four strings, specifying the following information:

- Session type name displayed in the list in the Change Session Types dialog box.
- Session type description displayed in the dialog box when the user selects the name.
- Default session startup command, passed the shell with the -c flag option as the startup command.
- Custom session startup command name, if a file of this name appears in the user's home directory, use it instead of the default session startup command.

EXAMPLES

The following is the resource file description used to create /mac/lib/sessiontypes/mac32, the A/UX Finder (32-bit) session type description file:

```
resource 'STR# (128) {
        {
```

```
                "A/UX Finder (32-bit)";
                "A/UX Finder (32-bit) is the normal"
                "session type.  Macintosh applications "
                "that are not 32-bit clean will "
                "not run properly in this mode.";

                "/mac/bin/mac32";
                ".mac32"
                }
        };
```

Resource file descriptions of this type can be compiled using the following command:

```
/mac/bin/rez -i /:mac:lib:rincludes types.r filename.r
```

FILES

`/mac/bin/Login`	`Login` executable
`/mac/bin/%Login`	`Login` resource file
`/mac/lib/sessiontypes`	Directory containing session type specification files
`/etc/shells`	List of standard shells
`/etc/utmp`	Accounting
`/etc/wtmp`	Accounting
`/etc/motd`	Message-of-the-day
`/etc/passwd`	Password file
`/etc/profile`	Systemwide profile for `sh`(1) and `ksh`(1)
`/etc/cshrc`	Systemwide profile for `csh`(1)
`$HOME/.profile`	Personal profile for `sh`(1) and `ksh`(1)
`$HOME/.login`	Personal profile used at login time for `csh`(1)
`$HOME/.cshrc`	Personal profile for `csh`(1)
`$HOME/.logout`	Personal profile used at logout for `csh`(1)
`/usr/mail/`*name*	Mailbox for user *name*

SEE ALSO

`init`(1M), `getty`(1M), `login`(1), `passwd`(1), `passwd`(4), `csh`(1), `ksh`(1), `sh`(1), `mail`(1), `newgrp`(1), `profile`(4), `environ`(5).

A/UX Essentials.

NAME

lpadmin — configure the lp spooling system

SYNOPSIS

/usr/lib/lpadmin -p*printer* [-c*class*] [-e*printer*] [-h] [-i*interface*] [-l] [-m*model*] [-r*class*] [-v*device*]

/usr/lib/lpadmin -x*dest*

/usr/lib/lpadmin -d[*dest*]

DESCRIPTION

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

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

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

-x*dest* Remove destination *dest* from the lp system. If *dest* is a printer and is the only member of a class, then the class is deleted, too. No other flag options are allowed with -x.

-p*printer* Name a printer to which all of the flag options below refer. If *printer* does not exist, then it is created.

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

-c*class* Insert printer *P* into the specified *class*, which is created if it does not already exist.

-e*printer* Copy the existing interface program of *printer* to be the new interface program for *P*.

-h Indicate that the device associated with *P* is hardwired. This flag option is assumed when creating a new printer, unless the -l flag option is sup-

plied.

-i*interface* Establish a new interface program for *P*, with *interface* being the pathname of the new program.

-l Indicate 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 enabling *P* again, its current *device* should be established using lpadmin.

-m*model* Select a model interface program for *P*, with *model* being one of the model interface names supplied with the lp software (see "Models" below).

-r*class* Remove the printer *P* from the specified *class*. If *P* is the last member of *class*, then *class* is removed.

-v*device* Associate a new *device* with printer *P*, with *device* being the pathname of a file that is writable by the lp administrator, lp. Note that there is nothing to stop an administrator from associating the same *device* with more than one *printer*. If only the -p and -v flag options are supplied, then lpadmin may be used while the scheduler is running.

Restrictions

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

Models

Model printer interface programs are supplied with the lp software. They are shell procedures that interface between lpsched and devices. All models reside in the directory /usr/spool/lp/model and may be used as they are with lpadmin -m. Models should have 644 permission if owned by lp and bin, or 664 permission if owned by bin and bin. Alternatively, lp administrators may modify copies of models and then use lpadmin -i to associate them with printers.

The following list describes the models and lists the options that they may be given on the lp command line by using the -o flag option:

`dumb` This model is the interface for a line printer without special functions and protocol. Form feeds are assumed. This is a good model to copy and modify for printers that do not have models.

`1640` This model is the interface for a Diablo 1640 printer using XON/XOFF protocol at 1200 baud. The options are:

`-12`
Specify 12-pitch. The default is 10-pitch.

`-f` Do not use the `450`(1) filter. The output has been preprocessed by either `450`(1) or the `nroff 450` driving table.

`hp` This model is the interface for a Hewlett-Packard 2631A line printer at 2400 baud. The options are:

`-c` Use compressed print.
`-e` Use expanded print.

`prx` This model is the interface for a Printronix P300 or P600 printer using the XON/XOFF protocol at 1200 baud.

EXAMPLES

Here are some examples of how to use the various printers:

1. Assuming there is an existing Hewlett-Packard 2631A line printer named `hp2`, it will use the `hp` model interface after the command:

```
/usr/lib/lpadmin -php2 -mhp
```

To obtain compressed print on `hp2`, use the command:

```
lp -dhp2 -o-c files
```

2. A Diablo 1640 printer named `st1` can be added to the `lp` 1 configuration with the command:

```
/usr/lib/lpadmin -pst1 -v/dev/tty20
-ml640
```

An `nroff` document may be printed on `st1` in any of the following ways:

```
nroff -T450 files | lp -dst1 -of
nroff -T450-12 files | lp -dst1 -of
nroff -T37 files | col | lp -dst1
```

The following command prints the password file on `st1` in 12-pitch:

```
lp -dst1 -o12 /etc/passwd
```

Note: The `-12` option of the `1640` model should never be used in conjunction with `nroff`.

FILES

`/usr/lib/lpadmin`
`/usr/spool/lp/*`
`/usr/lib/OUTQLCK`

SEE ALSO

`enable`(1), `lp`(1), `lpstat`(1), `nroff`(1), `accept`(1M), `lpsched`(1M).

NAME

lpc — line-printer control program

SYNOPSIS

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

DESCRIPTION

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

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

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

`? [ command ... ]`

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

`abort { all | printer ... }`
Terminate an active spooling daemon on the local host immediately and then disable printing for the specified printers. (To prevent new daemons from being started by lpr.)

`clean { all | printer ... }`
Remove any temporary files, data files, and control files that cannot be printed from the specified printer queue(s) on the local machine. (For example, do this so as not to form a complete printer job.)

`disable { all | printer ... }`
Turn off the spooling queues for the specified printers. This command prevents lpr from entering new printer jobs in the queue.

`down { all | printer } message ...`
Turn off the spooling queue for the specified printers, disable printing, and put `message` in the printer status file. The message doesn't need to be quoted. The remaining arguments are treated like `echo`(1). This command is normally used to take a printer down and let others know why. `lpq` indicates the printer is down and prints the status message.

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

`exit`

`quit`
Exit from `lpc`.

`restart { all | printer ... }`
Attempt to start a new printer daemon. This command is useful when some abnormal condition causes the daemon to die unexpectedly and leaves jobs in the queue. `lpq` reports that no daemon is present when this condition occurs. If the user is the super-user, try to abort the current daemon first, that is, kill and restart a stuck daemon.

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

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

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

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

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

FILES

/etc/printcap	Printer description file
/usr/spool/*	Spool directories
/usr/spool/*/lock	Lock file for queue control

SEE ALSO

lpd(1M), lpr(1), lpq(1), lprm(1), printcap(4).

DIAGNOSTICS

Here are some of the common error messages and a brief explanation of each:

?Ambiguous command
: The abreviation matches more than one command.

?Invalid command
: No match was found.

?Privileged command
: The command can be executed by root only.

NAME

lpd — 4.2 line-printer daemon

SYNOPSIS

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

DESCRIPTION

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

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

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

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

J Job name: string to be used for the job name on the banner page (the page with the job ID)

- `C` Classification: string to be used for the classification line on the banner page
- `L` Literal: the line containing identification info from the password file and causing the banner page to be printed
- `T` Title: string to be used as the title for `pr`(1)
- `H` Host name: name of the machine where `lpr` was invoked
- `P` Person: login name of the person who invoked `lpr` that is used to verify ownership by `lprm`
- `M` Mail: mail to be sent to the specified user when the current print job completes
- `f` Formatted file: the name of a file already formatted, which is to be printed
- `l` Similar to "`f`," but also passing control characters and not making page breaks
- `p` Filename: name of a file to be printed using `pr`(1) as a filter
- `t` Troff file: the file containing `troff`(1) output (`cat` phototypesetter commands)
- `n` Ditroff file: the file containing device-independent troff output
- `d` DVI file: the file containing `Tex`(l) output (DVI format from Stanford)
- `g` Graph file: the file containing data produced by `plot`(3X)
- `c` Cifplot file: the file containing data produced by `cifplot`
- `v` the file containing a raster image
- `r` the file containing text data with FORTRAN carriage-control characters
- `1` Troff Font R: the name of the font file to use instead of the default
- `2` Troff Font I: the name of the font file to use instead of the default
- `3` Troff Font B: the name of the font file to use instead of the default
- `4` Troff Font S: the name of the font file to use instead of the default

W Width: the number of characters to specify the page width used by pr(1) and the text filters

I Indent: the number of characters to indent the output by (in ASCII output)

U Unlink: the name of the file to be removed on completion of printing

N Filename: the name of the file that is being printed, or a blank for the standard input (when lpr is invoked in a pipeline)

If a file cannot be opened, a message is logged via syslog(3) by using the LOG_LPR facility. lpd tries up to 20 times to reopen a file it expects to be there, after which it skips the file to be printed.

lpd uses flock(2) to provide exclusive access to the lock file and to prevent multiple daemons from becoming active simultaneously. If the daemon should be killed or die unexpectedly, the lock file need not be removed. The lock file is kept in a readable ASCII form and contains two lines. The first line is the process ID of the daemon, and the second is the control filename of the current job being printed. The second line is updated to reflect the current status of lpd for the programs lpq(1) and lprm(1).

FILES

/etc/printcap	Printer description file
/usr/spool/*	Spool directories
/usr/spool/*/minfree	Minimum free space to leave
/dev/printer	Line-printer device
/dev/printer.socket	Socket for local requests
/etc/hosts.equiv	Lists of machine names allowed printer access
/etc/hosts.lpd	Lists of machine names allowed printer access, but not under same administrative control

SEE ALSO

lpc(1M), pac(1), lpr(1), lpq(1), lprm(1), syslog(3), printcap(4).

See lpsched(1M)

NAME

lpsched, lpshut, lpmove — start or stop the LP request scheduler and move requests

SYNOPSIS

/usr/lib/lpsched

/usr/lib/lpshut

/usr/lib/lpmove *requests dest*

/usr/lib/lpmove *dest1 dest2*

DESCRIPTION

lpsched schedules requests taken by lp(1) for printing on line printers.

lpshut shuts down the line printer scheduler. All printers that are printing at the time lpshut is invoked will stop printing. Requests that were printing at the time a printer was shut down will be reprinted in their entirety after lpsched is started again. All LP commands perform their functions even when lpsched is not running.

lpmove moves requests that were queued by lp(1) between LP destinations. This command may be used only when lpsched is not running.

The first form of the command moves the named *requests* to the LP destination, *dest*. *Requests* are request ids as returned by lp. The second form moves all requests for destination *dest1* to destination *dest2*. As a side effect, lp will reject requests for *dest1*.

Note that lpmove never checks the acceptance status (see accept(1M)) for the new destination when moving requests.

FILES

/usr/lib/lpsched
/usr/lib/lpshut
/usr/lib/lpmove
/usr/spool/lp/*

SEE ALSO

accept(1M), enable(1), lp(1), lpadmin(1M), lpstat(1).

See lpsched(1M)

NAME

lptest — generate line-printer ripple pattern

SYNOPSIS

lptest [*length* [*count*]]

DESCRIPTION

lptest writes the traditional "ripple test" pattern on standard output. In 96 lines, this pattern prints all 96 printable ASCII characters in each position. While originally created to test printers, lptest is quite useful for testing terminals, driving terminal ports for debugging purposes, or doing any other task where a quick supply of random data is needed.

The argument *length* specifies the output line length if the default length of 79 is inappropriate.

The argument *count* specifies the number of output lines to be generated if the default count of 200 is inappropriate. Note that if *count* is to be specified, *length* must be also be specified.

SEE ALSO

lpc(1M), lpd(1M), lpr(1), lpq(1), lprm(1).

NAME

macquery — post a Macintosh® alert box to query the user

SYNOPSIS

macquery [-t *timeout*] [-a] [-c] [-n] [-s] *resource-file* *alertID* [*parm1 ... parm4*]

DESCRIPTION

macquery is invoked by shell scripts or commands that do not directly use the Macintosh Toolbox but still need to present a Macintosh interface when asking the user simple questions or giving the user information. It is used to post an alert box that asks a question or supplies information. The alert box may include static text, icons, and QuickDraw™ pictures, but should not include check boxes, radio buttons, editable text fields, or other more complex controls. (See Chapter 13 of *Inside Macintosh, Volume I*, for more information about alerts and controls.) Typically, Apple's ResEdit utility is used to create alert resource definitions.

The command that invokes macquery must supply a resource filename *resource-file* and the ID *alertID* of an alert resource defined in that file. The alert is posted exactly as it is defined in the resource, unless a flag option indicates that one of the standard alert icons should also be displayed in the alert box. (See *Inside Macintosh, Volume I*, for more discussion of the various types of standard alerts.)

Up to four parameter strings may be included in the arguments. These replace the corresponding special strings '^0' through '^3' in the strings in the alert definition. (See the discussion of the ParamText call in *Inside Macintosh, Volume I*, for further discussion of parameter strings usage.)

When the user chooses an enabled button, macquery closes the alert and exits. The exit status indicates which button the user chose (or that an error occurred). See the section "EXIT CODES" for more information.

FLAG OPTIONS

The following flag options are interpreted by macquery:

-t *timeout*

Select the default button automatically if the user has not selected a button in the alert box after *timeout* seconds. The default button must be the first item in the alert definition and is indicated visually by a bold outline.

`-a` Post the alert box exactly as it is defined in the resource file.

`-c` Post the alert box as a Caution alert box.

`-n` Post the alert box as a Note alert box.

`-s` Post the alert box as a Stop alert box.

EXAMPLE

Because `fsck`(1M) is not linked as an A/UX® Toolbox command, the Macintosh environment does not have to be running to run `fsck`. However, `fsck` can invoke `macquery` to ask the user whether to repair damaged file systems. If the Macintosh environment is not running, `macquery` exits quickly with an error exit status. In this case, if `fsck` is running on `/dev/console`, its prompt messages would be displayed on the screen, and `fsck` reads the user's response from `/dev/console` instead of using the exit status of `macquery`.

The command that `fsck` uses to display the alert box is:

```
/mac/bin/macquery -ttimeout -c /etc/fsck 129 file-system-mount-point
```

In this example, the value of *timeout* is supplied as an argument to `fsck`. The *file-system-mount-point* is the pathname at which the file system that needs repairs would be mounted, for example, "/".

EXIT CODE

The exit status is used both to indicate an error and to report which button the user clicked.

If the exit status is 1, an error occurred. For example, the Macintosh environment is not running, or the resource file is missing. Diagnostic alert boxes or printed messages are posted.

If the exit status is between 201 and 225, the user chose the corresponding item numbered between 1 and 25 in the alert resource. Thus, an exit status of 201 indicates that the default button was chosen. Constants named ANSWER_MIN and ANSWER_MAX have been defined for these values in `/usr/include/apple/macquery.h` for use by C programmers.

FILES

```
/mac/bin/macquery
/mac/bin/%macquery
/usr/include/apple/macquery.h
```

SEE ALSO

Inside Macintosh, Volumes I and V.

See brc(1M)

NAME

mailq — list the contents of the mail queue

SYNOPSIS

mailq [-v]

DESCRIPTION

mailq lists the contents of the mail queue.

mailq interprets the following flag option:

-v Show all details of jobs in the mail queue.

FILES

/usr/ucb/mailq
/usr/spool/mqueue/* temp files

SEE ALSO

sendmail(1M).

NAME

makedbm — make a yellow pages dbm file

SYNOPSIS

makedbm [-i *yp-input-file*] [-o *yp-output-name*]
[-d *yp-domain-name*] [-m *yp-master-name*] *infile outfile*

makedbm [-u *dbmfilename*]

DESCRIPTION

makedbm takes *infile* and converts it to a pair of files in dbm(3X) format, namely *outfile*.pag and *outfile*.dir. Each line of the input file is converted to a single dbm record. All characters up to the first tab or space form the key, and the rest of the line is the data. If a line ends with \, then the data for that record is continued on to the next line. It is left for the clients of the yellow pages to interpret #; makedbm does not itself treat it as a comment character. *infile* can be -, in which case standard input is read.

makedbm is meant to be used in generating dbm files for the yellow pages, and it generates a special entry with *yp-last-modified*, which is the date of *infile* (or the current time, if *infile* is -).

FLAG OPTIONS

-i Create a special entry with the key YP_INPUT_FILE.

-o Create a special entry with the key YP_OUTPUT_NAME.

-d Create a special entry with the key YP_DOMAIN_NAME.

-m Create a special entry with the key YP_MASTER_NAME. If no master host name is specified, YP_MASTER_NAME will be set to the local host name.

-u Undo a dbm file. That is, print out a dbm file one entry per line, with a single space separating keys from values.

EXAMPLE

It is easy to write shell scripts to convert standard files such as /etc/passwd to the key value form used by makedbm. For example

```
#!/bin/awk -f
BEGIN { FS = ":"; OFS = "\t"; }
{ print $1, $0 }
```

takes the /etc/passwd file and converts it to a form that can be read by makedbm to make the yellow pages file passwd.byname. That is, the key is a username, and the value

is the remaining line in the /etc/passwd file.

FILES

/etc/yp/makedbm

SEE ALSO

yppasswd(1), dbm(3X).

NAME

mkfs — construct an SVFS file system

SYNOPSIS

/etc/mkfs *device-file blocks*[:*inodes*] [*gap modulus*]

/etc/mkfs *device-file proto* [*gap modulus*]

DESCRIPTION

mkfs constructs a System V file system (SVFS) by writing on the partition (logical disk) associated with *device-file* according to the directions found in the remainder of the command line. The command waits 10 seconds before starting to construct the file system. If the second argument is given as a string of digits (*blocks*), mkfs builds a file system with a single empty directory on it. The size of the file system corresponds to the value of *blocks* interpreted as a decimal number. This is the number of physical disk blocks the file system will occupy. The content of block 0 of the new file system is left uninitialized. If *inodes* is omitted, the default used is the value resulting when *blocks* is divided by 4.

If the second argument is a filename that can be opened (*proto*), mkfs treats it as a *proto* file containing specifications for controlling the creation of a new file system. The overall format of the *proto* file is as follows:

program
blocks inodes
file-system-mode user-id group-id
[*directory-name directory-mode user-id group-id*
[*file-name file-mode user-id group-id initial-contents*]
.
.
.
$]
.
.
.

(Braces surround optional items.)

In the first line of the *proto* file format, *program* should be replaced with the name of a file to be copied onto block 0 of the new file system. This collection of bytes is sometimes called the bootstrap program.

In the second line of the *proto* file format, *blocks* should be replaced with the size of the new file system in disk (512-byte) blocks. Typically the size is the number of blocks within a partition created with Apple® HD SC Setup. Refer to the disk management section of *A/UX Local System Administration* for details about the use of Apple HD SC Setup in terms of A/UX®.

Appearing after *blocks* in the second line is *inodes*. It should be replaced with the the number of inode slots for the new file system. Each inode slot can contain the operating system data that describes one file. The maximum number of configurable inode slots is 65,500, but the actual number depends on the number of blocks available to the new file system. Eight inode slots fill one disk block (512 bytes).

In the third line of the *proto* file format is *file-system-mode*. The first three characters of *file-system-mode* are `d--`. The last three characters of *file-system-mode* are the permission digits for the owner, group, and all other users that the mount point acquires whenever the new file system is mounted on it. See `mount`(1M) and `chmod`(1) for details.

Appearing after *file-system-mode* in the third line is *user-id*. It should be replaced with the numeric user ID of the user account that you wish to own the file. Appearing after *user-id* is *group-id*. It should be replaced with the numeric group ID of the group account that you wish to be associated with the file.

The optional fourth line of the *proto* file format is the beginning of a directory specification. A directory specification is unusual because it requires more than one line to specify completely. At least one other line is required, and it must contain the end-directory delimiter `$`. Between the starting and ending lines of a directory specification, you can place any number of file specifications or additional (nested) directory specifications. The number of lines in a directory specification depends on the number of files and nested directories with which the file system is to be initialized.

On the first line of a directory specification is *directory-name*, which should be replaced with a legal A/UX filename (up to 14 characters). Appearing after *directory-name* is *directory-mode*. Its replacement value can be treated the same as *file-system-mode*. Appearing after *directory-mode* is *user-id* and *group-id*, which have already been described.

The remaining lines of the directory specification are any number of file specifications, any number of embedded directory specifications, and lastly, a line containing the end-directory delimiter `$`.

The length of a file specification is one line. It begins with a value for *file-name*, which should be a legal A/UX filename (up to 14 characters).

Appearing after *filename* is *file-mode*. It should be replaced with a 6-character string, where the first character specifies the file type:

- `-` Specify a regular file.
- `b` Specify a block device file.
- `c` Specify a character device file.

See `mknod`(1M) for explanations of block and character device files.

The second character of *file-mode* specifies whether the set-user-ID permission is set or not:

- `u` Set the set-user-ID mode.
- `-` Do not set the set-user-ID mode.

The third character of *file-mode* indicates whether the set-group-ID permission is set or not:

- `g` Set the set-group-ID mode.
- `-` Do not set the set-group-ID mode.

The last three characters of *file-mode* are used to specify the octal number corresponding to the desired octal permission digits for the owner, group, and all other users. See `chmod`(1).

Appearing after *file-mode* are *user-id* and *group-id*, which have already been described.

Appearing after *group-id* is *initial-contents*. It should be replaced with the pathname of the file which will be used as the source of data that is copied into *file-name*.

If the file specification line is supposed to represent a device file, the file specification follows a slightly different format:

file-name file-mode user-id group-id major-no minor-no

As can be seen, *initial-contents* is replaced with major and minor device numbers. See `intro`(7) for more information about device files, and about major and minor device numbers.

Before any file specification can be given, an enclosing directory specification must be given. The format of a directory entry is similar to a file-specification line but lacks *initial-contents* information. For each directory specification, `mkfs` makes the directory entries . and .. before continuing. Once these directory provisions are made, it can build the files for any file-specification lines that might follow up to the end-directory delimiter. Since nested directory specifications are permitted, `mkfs` recursively builds those nested files and directories.

A sample *proto* file specification is:

```
/stand/diskboot
4872 110
d--777 3 1
usr    d--777 3 1
       sh     ---755 3 1 /bin/sh
       john   d--755 6 1
              $
       b0     b--644 3 1 0 0
       c0     c--644 3 1 0 0
       $
$
```

The following directory listings illustrate the initial contents of the file system that would follow from the preceding specification:

```
% ls -ld usr
drwxr-xr-x   3 sys      daemon         96 Aug  7 14:43 usr
% ls -lR usr
total 93
brw-r--r--   1 sys      daemon      0,  0 Aug  7 14:43 b0
crw-r--r--   1 sys      daemon      0,  0 Aug  7 14:43 c0
drwxr-xr-x   2 6        daemon         32 Aug  7 14:42 john
-rwxr-xr-x   1 sys      daemon      46172 Aug  7 14:42 sh

usr/john:
total 0
```

The files displayed for the `usr` directory are listed in reverse of the order of their creation because the `ls` command sorts each line alphabetically according to filename.

Whether or not a *proto* file is given, the rotational *gap* and the *modulus* values can be specified within the `mkfs` command line. The value of *gap* allows certain disk blocks to be treated as logically contiguous even though they are not physically contiguous. Specifically, those blocks that are *gap* blocks apart are treated as if they are contiguous during reads and writes. By doing this, the time delay between two consecutive reads or writes of blocks can be accounted for and the disk media does not rotate beyond the location of the next physical block. Rather than wait for the disk to make a complete revolution before the missed block comes under the read/write head once again, performance is better if alternating disk blocks are treated as if they were contiguous.

The value of *modulus* is needed to help determine what blocks are treated as logically contiguous. With each complete revolution, some extra offset may have to be introduced besides the fixed value *gap*. For example, for a *gap* value of 2 and a hypothetical 10 blocks per revolution, physical blocks 0, 2, 4, 6, and 8 would be treated as contiguous. However, if this were continued throughout the disk, odd-numbered physical blocks would never become accessible. A *modulus* value of 10 corrects the mapping so that after physical block 8 is mapped, block 1 is mapped, followed by 3, 5, 7, and 9.

Also note that modern hard disks perform a similar function internally so that the operating system need not be encumbered with the function of disk-block remapping. For all but a very few rare cases (hard disks not sold by Apple and of old vintage), these operating system facilities do not result in increased performance. Even when optimization is possible, it cannot be achieved unless you can determine from technical specifications for the disk what values are needed.

The default values `1` for *gap* and `1` for *modulus* suppress the remapping of disk blocks. The default values are used if *gap* and *modulus* are considered illegal values or if they are omitted.

EXAMPLE

To make an 800 KB file system on a 3.5-inch floppy disk, use

```
mkfs /dev/rfloppy0 1600
```

This makes a file system on the floppy media referenced through `/dev/rfloppy0`. The new file system extends for 1600 512-byte disk blocks (800 KB).

FILES

`/etc/fs/svfs/mkfs`

SEE ALSO

`chmod`(1), `fsirand`(1M), `mknod`(1M), `dir`(4), `fs`(4), `intro`(7), `boot`(8).

BUGS

When a *proto* file is used, `mkfs` can create a file system larger than the physical media.

If a *proto* file is used, it is not possible to initialize a file larger than 64 KB, nor is there a way to specify links.

NAME

mkfs1b — construct a file system with 512-byte blocks

SYNOPSIS

/etc/mkfs1b *special blocks*[: *inodes*] [*m n*]
/etc/mkfs1b *special proto* [*m n*]

DESCRIPTION

mkfs1b constructs a file system by writing on the special file *special*. mkfs1b operates exactly like mkfs except that the logical blocks created are 512 bytes instead of 1024 bytes.

In the first form of the command, a numeric size is given and mkfs1b builds a file system with a single empty directory on it. The number of inodes is calculated as a function of the file system size. *m* is an interleave factor for building the freelist and *n* is a modulo for *m*. See the example for usage.

Note: All file systems should have a lost+found directory for fsck(1M); this should be created for each file system by running mklost+found(1M) in the root directory of a newly created file system, after the file system is first mounted.

In bootstrapping, the second form of mkfs1b is sometimes used. In this form, the file system is constructed according to the directions found in the prototype file *proto*. The prototype file contains tokens separated by spaces or new lines. The first token is the name of a file to be copied onto sector zero as the bootstrap program. The second token is a number specifying the size of the created file system. Typically it will be the number of blocks on the device, perhaps diminished by space for swapping. The next token is the number of inodes in the i-list. The next set of tokens comprise the specification for the root file. File specifications consist of tokens giving the mode, the user ID the group ID, and the initial contents of the file. The syntax of the contents field depends on the mode.

The mode token for a file is a 6 character string. The first character specifies the type of the file. (The characters –bcd specify regular, block special, character special and directory files, respectively.) The second character of the type 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 three 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 is a pathname whence the contents and size are copied.

If the file is a block or character special file, two decimal number tokens follow which give the major and minor device numbers.

If the file is a directory, `mkfslb` makes the entries `.` and `..` and then reads a list of names and (recursively) file specifications for the entries in the directory. The scan is terminated with the token `$`.

A sample prototype specification follows:

```
/usr/mdec/uboot
4872 55
d--777 3 1
usr    d--777 3 1
       sh     ---755 3 1 /bin/sh
       ken    d--755 6 1
              $
       b0     b--644 3 1 0 0
       c0     c--644 3 1 0 0
       $
$
```

EXAMPLE

```
mkfslb /dev/fd0 2000 7 50
```

makes a file system in which 2000 is the total size of the file system to be put on `/dev/fd0`; 7 is a sector interleave number which is used to stagger the disk blocks for more rapid reading, every 7 blocks, and 50 is a modulo operator that forces the sector interlace number first to allocate all blocks in the first 50 sectors, then the next 50, etc.

Note: The proper selection of the *m* and *n* parameters can improve disk efficiency. Disks which have full or partial track buffering should specify a *m* and *n* of 1 and 1. *m* and *n* for other disks must be determined by trial and error as the disk latency is related to rotational latency and CPU speed.

FILES

/etc/mkfslb

SEE ALSO

fsck(1M), mklost+found(1M), dir(4).

BUGS

The default is 3500, which is probably not useful on any disk.
There should be some way to specify links.
There should be some way to specify bad blocks.
Should make lost+found automatically.

NAME

mklost+found — make a lost+found directory for fsck

SYNOPSIS

mklost+found

DESCRIPTION

A directory lost+found is created in the current directory and a number of empty files are created therein and then removed so that there will be empty slots for fsck(1M). This command should be run immediately after first mounting and changing directory to a newly created file system. For small file systems, it is sufficient (and much faster) to simply make a lost+found directory. Up to 30 files can be recovered in it.

EXAMPLE

```
mklost+found
```

in the current directory, creates a directory with empty slots named lost+found.

FILES

/bin/mklost+found

SEE ALSO

fsck(1M), mkfs(1M).

BUGS

Should be done automatically by mkfs.

NAME

mknod — build device file

SYNOPSIS

/etc/mknod *name type* [*major minor*]

/etc/mknod *name* p

DESCRIPTION

mknod makes a directory entry and corresponding inode for a device file. The first argument is the *name* of the entry. *type* can be "b" for block interfaces to devices; "c" for character or raw device interfaces; or "p" for named pipes. In the latter case (p), no other arguments are needed. Otherwise, the last two arguments (*major* and *minor*) are required. These numbers specify the *major* device type and *minor* device (e.g., unit, drive or line number), which may be either decimal or octal.

The assignment of major device numbers is specific to each system.

EXAMPLE

```
mknod /dev/tty4 c 3 4
```

would create file /dev/tty4 as a character device file with major number 3 and minor number 4.

FILES

/etc/mknod

SEE ALSO

mknod(2).

NAME

mkslipuser — initialize the slip user database

SYNOPSIS

/etc/mkslipuser

DESCRIPTION

mkslipuser is used to create the file /etc/slip.user based on the configuration file /etc/slip.config. The /etc/slip.user file records the current number of slip users on the system and the number of available slip interfaces. The /etc/slip.user file is not human readable. Use dslipuser(1M) to display the contents of the /etc/slip.user file.

Only the superuser may initialize the slip user database.

EXAMPLE

```
/etc/mkslipuser
```

FILES

/etc/slip.config
: System configuration file

/etc/slip.user
: User file to be created

/etc/slip.hosts
: User-to-host address mapping file (not used directly but of related interest)

SEE ALSO

dslipuser(1M), slip(1M), slip.config(4), slip.user(4).

NAME

module_dump — identify configuration information stored within the named kernel file

SYNOPSIS

module_dump *kernel*

DESCRIPTION

module_dump dumps information from the section MODULES in the A/UX kernel specified by the argument *kernel*. The *kernel* is the filename of the A/UX kernel you want information from. For example,

```
module_dump /unix
```

This is information normally put into place by the autoconfiguration process and describes the environment for which the kernel is configured.

FILES

/usr/bin/module_dump

SEE ALSO

autoconfig(1M).

monacct(1M)

See acctsh(1M)

NAME

mount, umount — mount and dismount file systems

SYNOPSIS

/etc/mount [-p]

/etc/mount -a [frv] [-t *type*] [-T *type*]

/etc/mount [-frv] [-t *type*] [-T *type*] [-o *options*] *device-file mount-point*

/etc/umount [-v] -h *host*

/etc/umount -a[v]

/etc/umount [-v] [*device-file*]...

/etc/umount [-v] [*mount-point*]...

DESCRIPTION

mount enables access to the files and directories in a file system contained in the disk or disk partition referenced as *device-file*. The topmost directory of the add-on file system is attached to the directory tree at *mount-point*. The directory *mount-point* must already exist. It serves as the entry point for the newly-mounted file system for as long as the file system remains mounted. If any files or directories had been placed below the *mount-point* directory before mounting, they become hidden. If *device-file* is specified as *host*:*mount-point* the file system type is assumed to be a network file system (NFS) that can be reached across the Ethernet network (see exports(4)).

umount disables access to the files and directories in an add-on file system referenced through *mount-point* or *device-file*.

mount and umount maintain a table of mounted file systems in /etc/mtab, described in mtab(4). If invoked without an argument, mount displays the table. Note that since /etc/mtab can be modified by commands other than mount and umount, its contents may not accurately reflect what is actually mounted. If invoked with only *device-file* or *mount-point*, mount searches /etc/fstab for an entry whose *device-file* or *mount-point* field matches the given argument. For example,

```
mount /usr
```

and

```
mount /dev/floppy0
```

are shorthand for

```
mount /dev/floppy0 /usr
```

if the following line is in /etc/fstab:

```
/dev/floppy0 /usr 5.2 rw 1 1
```

MOUNT FLAG OPTIONS

The following flag options are interpreted by mount:

-p Print the list of mounted file systems in a format suitable for use in /etc/fstab.

-a Attempt to mount all the file systems described in /etc/fstab. In this case, *device-file* and *mount-point* are taken from /etc/fstab. If a type is specified with the -t or -T options, all of the file systems in /etc/fstab with that type are mounted. File systems are not necessarily mounted in the order listed in /etc/fstab.

-f Fake a new /etc/mtab entry. This does not actually mount any file systems.

-v Provide verbose output; mount displays a message indicating which file system is being mounted.

-t or -T The next argument is the file system type. The accepted types are: 4.2 (UFS), 5.2 (SVFS), nfs, and pc; see fstab(4) for a description of the legal file system types.

-o Specifies *options*, a list of comma-separated words from the following list. Some options are valid for all file system types, while others apply to a specific type only.

options valid on all file systems (the default is rw, noquota) are:

quota	Enforce usage limits.
noquota	Do not enforce usage limits.
rw	Read/write.
ro	Read-only.
suid	Allow set-user-ID execution.
nosuid	Do not allow set-user-ID execution.

options specific to `nfs` (NFS) file systems are:

`bg`	If the first mount attempt fails, retry in the background.
`fg`	Retry in the foreground.
`retry=`*n*	Set number of mount failure retries to *n*.
`rsize=`*n*	Set read buffer size to *n* bytes.
`wsize=`*n*	Set write buffer size to *n* bytes.
`timeo=`*n*	Set NFS timeout to *n* tenths of a second.
`retrans=`*n*	Set number of NFS retransmissions to *n*.
`port=`*n*	Set server IP port number to *n*.
`soft`	Return error if server doesn't respond.
`hard`	Retry request until server responds.

The defaults are

```
fg,retry=1,timeo=7,retrans=4,port=NFS_PORT,hard
```

with defaults for `rsize` and `wsize` set by the kernel.

The `bg` option causes `mount` to run in the background if `mountd`(1M) of the server does not respond. `mount` attempts each request `retry=`*n* times before giving up. Once the file system is mounted, each NFS request made in the kernel waits `timeo=`*n* tenths of a second for a response. If no response arrives, the timeout is multiplied by 2, and the request is retransmitted. When `retrans=`*n* retransmissions have been sent with no reply, a `soft` mounted file system returns an error on the request, and a `hard` mounted file system retries the request. File systems that are mounted `rw` (read/write) should use the `hard` option. The number of bytes in a read or write request can be set with the `rsize` and `wsize` options.

`-r` Mount the specified file system read-only. This is a shorthand for

```
mount -o ro device-file mount-point
```

Physically write-protected and magnetic-tape file systems must be mounted read-only, or errors will occur when access times are updated, whether or not any explicit write is attempted.

UMOUNT FLAG OPTIONS

The following flag options are interpreted for unmounting a file system:

-h Unmount all file systems listed in `/etc/mtab` that are remotely mounted from *host*.

-a Attempts to unmount all the file systems currently mounted (listed in `/etc/mtab`). In this case, *device-file* is taken from `/etc/mtab`.

-v Provide verbose output; `umount` displays a message indicating the file system being unmounted.

EXAMPLES

```
mount /dev/dsk/c0d0s2 /usr
```

mounts a local disk.

```
mount -at 5.2
```

mounts all System V.2 file systems.

```
mount -t nfs serv:/usr/src /usr/src
```

mounts remote file system.

```
mount serv:/usr/src /usr/src
```

mounts remote file system.

```
mount -o hard serv:/usr/src /usr/src
```

mounts remote file system, but with hard mount.

```
mount -p > /etc/fstab
```

saves current mount table state in `/etc/fstab` where it will continue to be available to assist with the automatic mounting and unmounting of file systems.

FILES

```
/etc/mount
/etc/umount
/etc/mtab
/etc/fstab
```

SEE ALSO

`mountd`(1M), `nfsd`(1M), `fsmount`(2), `unmount`(2), `mount`(3), `umount`(3), `fstab`(4), `mtab`(4).

BUGS

Mounting file systems full of garbage crashes the system.

If the directory on which a file system is to be mounted is a symbolic link, the file system is mounted on the directory to which the symbolic link refers, rather than is mounted on top of the symbolic link itself.

NAME

mountd — NFS mount request server

SYNOPSIS

/usr/etc/rpc.mountd

DESCRIPTION

mountd is an RPC server that answers file system mount requests. It reads the file /etc/exports, described in exports(4), to determine which file systems are available to which machines and users. It also provides information as to which clients have file systems mounted. This information can be printed using the showmount(1M) command.

The mountd daemon is normally invoked from /etc/inittab.

FILES

/usr/etc/rpc.mountd

SEE ALSO

showmount(1M), exports(4), services(4), inittab(4).

NAME

named — Internet domain name server

SYNOPSIS

named [-d *debuglevel*] [-p *port#*] [*bootfile*]

DESCRIPTION

named is the Internet domain name server. Without any arguments, named will read the default boot file /etc/named.boot, read any initial data and listen for queries.

Flag options are:

-d Print debugging information. A number after the "d" determines the level of messages printed.

-p Use a different port number. The default is the standard port number as listed in /etc/services.

Any additional argument is taken as the name of the boot file. The boot file contains information about where the name server is to get its initial data. The following is a small example:

```
;
;   boot file for name server
;
; type       domain       source file or host
;
domain       berkeley.edu
primary      berkeley.edu     named.db
secondary  cc.berkeley.edu 10.2.0.78 128.32.0.10
cache                       named.ca
```

The first line specifies that "berkeley.edu" is the domain for which the server is authoritative. The second line states that the file "named.db" contains authoritative data for the domain "berkeley.edu." The file "named.db" contains data in the master file format described in RFC883 except that all domain names are relative to the origin; in this case, "berkeley.edu" (see below for a more detailed description). The second line specifies that all authoritative data under "cc.berkeley.edu" is to be transferred from the name server at 10.2.0.78. If the transfer fails it will try 128.32.0.10 and continue trying the address, up to 10, listed on this line. The secondary copy is also authoritative for the specified domain. The fourth line specifies data in "named.ca" is to be placed in the cache (like well known data such as locations of root domain

servers). The file "named.ca" is in the same format as "named.db."

The master file consists of entries of the form:

$INCLUDE *<filename>*
$ORIGIN *<domain>*
<domain> <opt_ttl> <opt_class> <type> <resource_record_data>

where *domain* is "." for root, "@" for the current origin, or a standard domain name. If *domain* is a standard domain name that does not end with ".", the current origin is appended to the domain. Domain names ending with "." are unmodified. The *opt_ttl* field is an optional integer number for the time-to-live field. It defaults to zero. The *opt_class* field is the object address type; currently only one type is supported, IN, for objects connected to the DARPA Internet. The *type* field is one of the following tokens; the data expected in the *resource_record_data* field is in parentheses.

A	a host address (dotted quad)
NS	an authoritative name server (domain)
MX	a mail exchanger (domain)
CNAME	the canonical name for an alias (domain)
SOA	marks the start of a zone of authority (5 numbers)
MB	a mailbox domain name (domain)
MG	a mail group member (domain)
MR	a mail rename domain name (domain)
NULL	a null resource record (no format or data)
WKS	a well know service description (not implemented yet)
PTR	a domain name pointer (domain)
HINFO	host information (cpu_type OS_type)
MINFO	mailbox or mail list information (request_domain error_domain)

NOTES

The following signals have the specified effect when sent to the server process using the kill(1) command.

SIGHUP	Causes server to read named.boot and reload database.
SIGINT	Dumps current data base and cache to /usr/tmp/named_dump.db
SIGUSR1	Turns on debugging; each SIGUSR1 increments debug level.
SIGUSR2	Turns off debugging completely.

FILES

/etc/named	
/etc/named.boot	name server configuration boot file
/etc/named.pid	the process ID
/usr/tmp/named.run	debug output
/usr/tmp/named_dump.db	dump of the name servers database

SEE ALSO

kill(1), gethostbyname(3N), signal(3), resolver(3), resolver(4).

NAME

ncheck — locate the filename associated with an i-node

SYNOPSIS

/etc/ncheck [-a] [-i *i-node-numbers*] [-s] [-T*file-system-type*] [*file-system*]

DESCRIPTION

ncheck with no argument generates a pathname and i-node list of all files on a set of default file systems. Names of directory files are followed by /.. The -i option reduces the report to only those files whose i-nodes follow. The -a option allows printing of the names . and .., which are ordinarily suppressed. The -s option reduces the report to special files and files with set-user-ID mode; it is intended to discover concealed violations of security policy.

The -T flag option indicates the file-system type, for example, 4.2 or 5.2. If this option is not present, ncheck attempts to determine the file-sytem type.

A file system may be specified.

EXAMPLE

```
ncheck /dev/rdsk/c0d0s0
```

reports the pathnames and i-nodes of files on the specified device.

BUGS

The report is in no useful order and probably should be sorted.

FILES

/etc/ncheck

SEE ALSO

fs(4), fsck(1M), sort(1).

DIAGNOSTICS

When the file-system structure is improper, ?? denotes the "parent" of a parentless file, and a pathname beginning with . . . denotes a loop.

NAME

ncstats — display kernel name cache statistics

SYNOPSIS

```
ncstats
```

DESCRIPTION

ncstats prints the contents of the kernel's name cache statistics structure, giving the number and percentage of each event. The statistics kept are:

hits	number of cache hits
misses	number of cache misses
long_enter	number of attempts to enter a long name (more than 32 bytes) in the cache
long_look	number of attempts to look up a long name
lru_empty	number of times the LRU list was empty
purges	number of times the entire cache was purged

EXAMPLE

```
ncstats
```

results in output similar to:

```
Directory name cache statistics:
   hits:        140144     (72%)
   misses:       53250     (28%)
   long_enter:    1078     ( 1%)
   long_look:     1126     ( 1%)
   lru_empty:        0     ( 0%)
   purges:        2471     ( 1%)
```

FILES

```
/etc/ncstats
```

BUGS

The percentages given don't always add up to 100%.

NAME

newaliases — rebuild the database for the mail aliases file

SYNOPSIS

newaliases

DESCRIPTION

newaliases rebuilds the random access database for the mail aliases file /usr/lib/aliases. It must be run each time /usr/lib/aliases is changed in order for the change to take effect.

FILES

/usr/ucb/newaliases

SEE ALSO

aliases(4).

NAME

newconfig — prepare and configure a new kernel

SYNOPSIS

/etc/newconfig [–v] [nonet] [*module*]... [no*module*]...

DESCRIPTION

newconfig is an A/UX® shell script that invokes the necessary commands to generate a new kernel. newconfig begins by calling newunix(1M) to install or remove the appropriate scripts and driver object files needed by autoconfig(1M). newconfig then invokes autoconfig to link the new kernel. The autoconfig scripts may ask the user for system options. Finally, newconfig invokes the shell script /etc/startup, which is produced by autoconfig. /etc/startup asks the user for any additional system configuration information. The new configuration does not take effect until the user reboots the system.

newconfig interprets the following flag option:

–v Causes newconfig to display its commands as they are invoked and to invoke those commands verbosely.

The keyword nonet causes the removal of TCP/IP networking capabilities.

Specification of *module* causes the named module to be configured in the resulting kernel, while no*module* causes the named module to be removed from the resulting kernel. The placeholder *module* is case-insensitive. More than one module can be specified on the command line. The modules listed below are supported by Apple®. Third-party vendors may provide additional modules.

appletalk
: AppleTalk® networking capabilities that include both LocalTalk™ and EtherTalk™.

bnet
: B-NET (Berkeley Networking) TCP/IP capabilities.

nfs
: Network file system (NFS) capabilities that automatically include B-NET.

debugger
: A/UX kernel debugger support.

`slip`
Serial line/internet protocol (SL/IP) capabilities.

`snd`
Enable use of the Macintosh sound chip.

`svfs`
System V file system (SVFS) capabilities.

`tc` Enable use of the Apple Tape Backup 40SC device.

`ufs`
Berkeley 4.2 file system (UFS) capabilities.

The following modules are usually included in conjunction with another module and are indirectly installed in or removed from the system. Normally, these modules are not explicitly specified by the user.

`ae6`
Apple EtherTalk Interface Card driver.

`at_atp`
AppleTalk ATP (AppleTalk Transaction Protocol) driver.

`at_papd`
AppleTalk PAP (Printer Access Protocol) driver.

`at_sig`
AppleTalk driver.

`atp`
AppleTalk driver.

`bnet_dr`
B-NET sockets code.

`ddp`
AppleTalk DDP (Datagram Delivery Protocol) driver.

`elap`
EtherTalk LAP (Link Access Protocol) driver.

`llap`
LocalTalk LAP driver.

`nfs_dr`
Network file system (NFS) driver.

`slots`
Enable calling the Slot Library.

`toolbox`
A/UX user interface device driver.

EXAMPLES

To prepare a kernel for TCP/IP services with NFS and AppleTalk networking services, type

```
/etc/newconfig nfs appletalk
```

To prepare a kernel that supports basic TCP/IP services, AppleTalk networking, the Macintosh sound chip, and Apple Tape Backup 40SC device, type

```
/etc/newconfig bnet appletalk snd tc
```

To prepare a kernel from which all networking services are removed, type

```
/etc/newconfig nonet noappletalk
```

FILES

`/etc/newunix`	A shell script called by `newconfig`
`/etc/autoconfig`	An executable utility called by `newconfig`
`/etc/startup`	A shell script created by `autoconfig`

SEE ALSO

`newunix`(1M), `autoconfig`(1M).

NAME

newfs — construct a new UFS file system

SYNOPSIS

/etc/newfs [–v] [*options*] *device-file*

DESCRIPTION

newfs constructs a Berkeley 4.2 file system (UFS) on *device-file*, which is the device-file on which the new file system is to be created. The placeholder *type* indicates the disk type; this *type* is used to find the appropriate disk name entry in /etc/disktab. The newfs command consults the disk label for disk partition information and /etc/disktab for disk architecture information, calculates the appropriate parameters to use in calling mkfs, and then builds the file system by invoking mkfs.

If the -v option is supplied, newfs prints out its actions, including the parameters passed to mkfs .

newfs uses fsirand as a security precaution.

Options that may be used to override default parameters passed to mkfs are:

-s *size*
: Specify the size of the file system in sectors. If this option is not present, the size information from the disk partition map will be used. See dpme(4).

-b *block-size*
: Specify the block size of the file system in bytes. The default value is 4096.

-f *frag-size*
: Specify the fragment size of the file system in bytes. The default value is 1024.

-t *tracks-per-cylinder*
: Specify the number of tracks per cylinder, which is equivalent to the number of heads on the disk drive. If this option is not present, the information from /etc/fs/ufs/disktab is used.

-c *cylinders-per-group*
: Specify the number of cylinders per cylinder group in a file system. The default value is 16.

-m *free-space*
: Specify the percentage of space reserved from use by normal

users. This value is known as the free-space threshold for the file system. The default value is 10%. This value can be changed later using `tunefs`(1M).

`-r` *revolutions-per-minute*
Specify the speed of the disk in revolutions per minute (usually 3600).

`-i` *number of bytes per inode*
Specify the density of inodes in the file system. The default is to create an inode for each 2048 bytes of data space. If few inodes are desired, a larger *bytes-per-inode* should be specified. If many inodes are desired, a smaller *bytes-per-inode* should be specified.

FILES

`/etc/fs/ufs/newfs`
Actually builds the file system.

SEE ALSO

`dp`(1M), `fsck`(1M), `fsirand`(1M), `tunefs`(1M), `dpme`(4), `disktab`(4), `ufs`(4), `gd`(7).

NAME

newunix — prepare for new kernel configuration

SYNOPSIS

/etc/newunix [[no]*module*] ...

DESCRIPTION

newunix is typically called by newconfig, but can also be used directly. However called, it begins the process of configuring a new kernel by installing (or removing) the appropriate scripts and driver object files needed by autoconfig. When you invoke it directly, you should run autoconfig afterwards to complete the kernel-configuration process. When you invoke it indirectly by running newconfig, autoconfig is called automatically, making newconfig easier to use.

The configuration of the new kernel is controlled by the arguments of *module*. Multiple invocations of newunix can be used to accumulate the new kernel configuration or to remove previously established modules through the argument format of no*module* (a couple of exceptions are noted in the following).

Possible values of *module* are:

appletalk	Provide AppleTalk® support.
bnet	Use basic networking. To turn off basic networking, use nonet rather than the expected nobnet.
nfs	Use the network file system (NFS). To turn off NFS support, use nonet rather than the expected nonfs.
slip	Provide support for the Serial Line/Internet Protocol (SL/IP).
tc	Provide support for the Apple® Tape Backup 40SC device.
toolbox	Use the A/UX® Toolbox.

EXAMPLES

To prepare the system for an NFS kernel, enter

```
/etc/newunix nfs
```

To prepare the system for a kernel that supports the tape controller, enter

```
/etc/newunix tc
```

If, after requesting tape controller support, you decide to remove it, enter

```
/etc/newunix notc
```

The three preceding examples can also be considered one example of staging a new kernel configuration, where the last two module requested, `tc` and `notc`, canceled each other out.

To proceed with the building of a new kernel for the currently accumulated configuration, run `autoconfig` as follows:

```
autoconfig -I -S /etc/startup
```

See `autoconfig`(1M) for complete details on running `autoconfig`. See `newconfig`(1M) for a more automatic way of preparing a new kernel configuartion.

FILES

`/etc/boot.d/*`	Driver object files
`/etc/install.d/*`	Installation scripts
`/etc/master.d/*`	Script files
`/etc/startup.d/*`	Startup programs
`/etc/uninstall.d/*`	Removal scripts
`/etc/init.d/*`	Initialization scripts

SEE ALSO

`autoconfig`(1M), `finstall`(1M), `newconfig`(1M).

"Installing and Administering AppleTalk," in *A/UX Local System Administration.*

NAME

nfsd, biod — NFS daemons

SYNOPSIS

/etc/nfsd [*nserver*...]

/etc/biod [*nserver*...]

DESCRIPTION

nfsd starts the NFS server daemons that handle client file system requests. *nservers* is the number of file system request daemons to start. This number should be based on the load expected on this server; four is a good number. If *nservers* is not specified it defaults to one.

biod starts *nservers* asynchronous block I/O daemons. This command is used on a NFS client to handle read-ahead and write-behind of buffer cached blocks. A good value for *nservers* is four; if not specified it defaults to one.

FILES

/etc/nfsd
/etc/biod

SEE ALSO

mountd(1M), exports(4).

NAME

nfsstat — Network File System statistics

SYNOPSIS

nfsstat [-csnrz]

DESCRIPTION

nfsstat displays statistical information about the Network File System (NFS) and Remote Procedure Call (RPC) interfaces to the kernel. It can also be used to reinitialize this information. If no flag options are given, the default is

```
nfsstat -csnr
```

That is, print everything and reinitialize nothing.

FLAG OPTIONS

The following flag options are interpreted by nfsstat:

-c Display client information. Only the client side NFS and RPC information will be printed. Can be combined with the -n and -r flag options to print client NFS or client RPC information only.

-s Display server information. Works like the -c flag option above.

-n Display NFS information. NFS information for both the client and server side will be printed. Can be combined with the -c and -s flag options to print client or server NFS information only.

-r Display RPC information. Works like the -n flag option above.

-z Zero (reinitialize) statistics. Can be combined with any of the above flag options to zero particular sets of statistics after printing them. The user must have write permission on /dev/kmem for this flag option to work.

FILES

/usr/etc/nfsstat	
/unix	system namelist
/dev/kmem	kernel memory

See acctsh(1M)

NAME

pac — gathers printer/plotter accounting information

SYNOPSIS

/etc/pac [−P*printer*] [−p*price*] [−s] [−r] [−c] [−m] [*name*]...

DESCRIPTION

pac reads the printer/plotter accounting files, accumulating the number of pages (the usual case) or feet (for raster devices) of paper consumed by each user, and printing out how much each user consumed in pages or feet and dollars. If any *names* are specified, then statistics are only printed for those users; usually, statistics are printed for every user who has used any paper.

FLAG OPTIONS

The following flag options are interpreted by pac:

−P*printer*
: Does accounting for the named printer. Normally, accounting is done for the default printer (site-dependent) or the value of the environment variable PRINTER is used.

−p*price*
: Uses the value *price* for the cost in dollars instead of the default value of 0.02 or the price specified in /etc/printcap.

−c Sorts the output by cost. Usually the output is sorted alphabetically by name.

−r Reverses the sorting order.

−s Summarizes the accounting information on the summary accounting file. This summarizing is necessary because the accounting file can grow by several lines per day on a busy system.

−m Ignores the host name in the accounting file so a user on multiple machines can have all of his or her printing charges grouped together.

FILES

/usr/adm/?acct	Raw accounting files
/usr/adm/?_sum	Summary accounting files
/etc/printcap	Printer-capability database

SEE ALSO

printcap(4).

BUGS

The relationship between the computed price and reality is as yet unknown.

NAME

ping — exercise the network by sending test packets to a named host

SYNOPSIS

/usr/etc/ping *host* [*timeout*]

DESCRIPTION

ping repeatedly sends an icmp echo packet to *host* and reports whether or not a reply was received. It keeps trying until *timeout* seconds have elapsed, or an answer is received. The default timeout is 20 seconds. The *host* argument can be a name or an internet address.

ping continues to send icmp echo packets to *host* and reports back until the process is killed. Use the interrupt character (usually CONTROL-C) to stop the output of ping once the packets are being returned. ping then prints statistics and exits.

FILES

/usr/etc/ping

SEE ALSO

icmp(5P).

NAME

pname — associate named partitions with device files

SYNOPSIS

/bin/pname [-a] [-c *controller*] [-d *disk*] [-s *slice*] [-t *type*] *name*

/bin/pname [-p]

/bin/pname -a[v]

/bin/pname -u *device-file* [*device-file*...]

DESCRIPTION

pname enables the system to recognize the partition with name *name*. The partition, *name*, must already exist (for creating partitions see dp(1M) and the description of HD SC Setup in *A/UX Local System Administration*).

pname maintains a table of partitions it has recognized in /etc/ptab, described in ptab(4). If invoked without an argument, pname displays all the partitions it has recognized in terms of the device files in /dev/rdsk.

When pname is invoked without an argument certain slices may not be reported as recognized, but can be honored nevertheless. This is the case when the associated partitions are named Root or Root&Usr (both on slice 0), Swap (on slice 1), or Usr (on slice 2) and when they are referenced appropriately as one of the following device files:

```
/dev/dsk/cnd0s0
/dev/dsk/cnd0s1
/dev/dsk/cnd0s2
```

When invoked to recognize a specific partition, pname writes to standard output the pathname of the device file that has been associated with the specified partition.

FLAG OPTIONS

The following flag options are interpreted by pname:

-p Prints the list of recognized partitions in a format suitable for use in /etc/ptab.

-a This option takes on two different meanings dependent upon the command usage. If pname is invoked with a partition name, this option will cause an entry (if one is not already present) for the partition to be added to /etc/ptab. If no

partition name is specified, pname will attempt to recognize all the partitions described in /etc/ptab. (In this case, *name*, *type*, *controller*, *disk*, and *slice* are taken from /etc/ptab.) Partitions are recognized in the order listed in /etc/ptab.

-v Verbose: pname displays a message indicating that the partition is being recognized.

-c This option is used to specify that this partition, *name*, resides on controller number *controller*. If this option isn't specified, pname will assume the controller number is zero.

-d This option is used to specify that the partition, *name*, resides on disk number *disk*. If this option isn't specified, pname will assume the disk number is zero.

-s This option is used to specify *slice* as the number by which the partition, *name*, will be recognized. If this option isn't specified, pname will choose an unused slice number. If the device corresponding to the slice number does not exist and the controller and disk numbers are valid, the device will be created in /dev/dsk and /dev/rdsk.

-t If this option is specified, *type* will be used as the type of the partition, instead of the default, Apple_UNIX_SVR2.

-u This option will cause pname to disassociate partitions with the specified devices.

EXAMPLES

`pname -a`	recognize all partitions
`pname -c1 "PeterC's part"`	recognize named partition on controller 1
`pname -p > /tmp/pstate`	save current partition state

FILES

/bin/pname
/etc/ptab
/dev/dsk/c[0-7]d[0-7]s*
/dev/rdsk/c[0-7]d[0-7]s*

SEE ALSO

dp(1M), mknod(1M), getptabent(3), ptab(4).

WARNINGS

Some of the actions that may be performed by pname require read and write permission on certain directories and certain files that may not be readable or writable to all users. For example, when invoked to display a list of all recognized partitions, pname will silently ignore all partitions associated with devices that are not readable to the user that invoked pname.

BUGS

The current revision of the software will not support colons (:) in partition names or partition types.

NAME

portmap — DARPA port to RPC program number mapper

SYNOPSIS

/etc/portmap

DESCRIPTION

portmap is a server that converts RPC program numbers into DARPA protocol port numbers. It must be running in order to make RPC calls.

When an RPC server is started, it will tell portmap what port number it is listening to, and what RPC program numbers it is prepared to serve. When a client wishes to make an RPC call to a given program number, it will first contact portmap on the server machine to determine the port number where RPC packets should be sent.

Normally, standard RPC servers are started from /etc/inittab.

FILES

/etc/portmap

SEE ALSO

rpcinfo(1M).

BUGS

If portmap crashes, all servers must be restarted.

NAME

powerdown — power down the system

SYNOPSIS

/etc/powerdown

DESCRIPTION

powerdown flushes the internal system buffers and powers down the machine.

FILES

/etc/powerdown

SEE ALSO

shutdown(1M), reboot(1M).

See brc(1M)

See acctsh(1M)

See acctsh(1M)

See acctsh(1M)

See transcript(1M)

See `transcript`(1M)

See transcript(1M)

See transcript(1M)

NAME

pstat — print system facts

SYNOPSIS

pstat [-p [-a]] [-b] [-i] [-m] [-n*namelist*] [-r*rate*] [-t] [-u*address*] [-v [*file*]

DESCRIPTION

pstat interprets the contents of certain system tables. If *file* is given, the tables are sought there, otherwise in /dev/kmem. Unless the -n flag option is used, the required namelist is taken from /unix.

FLAG OPTIONS

The following flag options are interpreted by pstat:

-a Under -p, describe all process slots rather than just active ones.

-b Print the system I/O buffer header information with the following headings:

LOC The core location of the buffer header

FLAGS Miscellaneous state variables encoded thus:

R The buffer is to be read.

W The buffer is to be written.

D The I/O is done.

E An error occurred during the I/O operation of the buffer.

B The buffer is busy.

P The buffer is being used for physical (raw) I/O.

M The buffer has map space allocated (not all machines).

W A process wants to access the buffer and is waiting for it.

A The buffer has aged.

Y The buffer is doing an asynchronous operation (the process that started the I/O does not wait for it to complete).

L The buffer contents have changed, and they need to be written out before the buffer can be reallo-

cated.

O The open routine has been called for this device.

S The buffer is "stale."

DEVICE The major and minor device numbers for the device to which the buffer is queued (or contains information from)

ADDR The core address of the data in the buffer

BLKNO The block number of the block on DEVICE.

-f Print the open file table with these headings:

LOC The core location of this table entry

FLG Miscellaneous state variables encoded thus:

R Open for reading

W Open for writing

P Pipe

CNT Number of processes that know this open file

INO The location of the inode table entry for this file

OFFS The file offset (see lseek(2))

-i Print the inode table with these headings:

LOC The core location of this table entry

FLAGS Miscellaneous state variables encoded thus:

L Locked.

U The update time fs(4) must be corrected.

A The access time must be corrected.

M The file system is mounted here.

W Wanted by another process (L flag is on).

T Contains a text file.

C Changed time must be corrected.

CNT Number of open file table entries for this inode

DEVICE Major and minor device number of file system in which this inode resides

INO Inumber within the device

MODE Mode bits (see chmod(2))

NLK Number of links to this inode

UID User ID of owner

SIZ/DEV Number of bytes in an ordinary file, or major and minor device of device-file

LOCK Address of the locklist structure for this inode

-m Print information about core memory allocation and a dump of the memory free map with these headings:

LOC The core address of the map entry

ADDR The "click" address of the area this entry refers to

SIZE The size of this area in "clicks"

-n*namelist*
Specify a namelist (system code file) other than the default of /unix.

-p Print the process table for active processes with these headings:

LOC The core location of this table entry

S Run state encoded thus:

0 No process

1 Waiting for some event

2 Runnable

3 Being terminated

4 Stopped under trace

5 Being created

6 Running

7 Being xswapped

F Flags (octal and additive) associated with the process:

0 Swapped

1 System process

2 Being traced by another process

4 Another tracing flag

10 Process cannot be woken by a signal

20 In core

40 Locked in memory

PRI Scheduling priority (see nice(2))

SIGNAL Signals received (signals 1-16 coded in bits 0-15)

UID Real user ID

TIM Time resident in seconds; times over 127 coded as 127

CPU Weighted integral of CPU time, for scheduler

NI Nice level (see nice (2))

PGRP Process number of root of process group (the opener of the controlling terminal)

PID The process ID number

PPID The process ID of parent process

ADDR If in core, the physical address of the page tables in the proc structure for the "u-area" of the process; if swapped out, the position in the swap area measured in multiples of 512 bytes

SIZE Size of process image in multiples of logical page size

WCHAN Wait channel number of a waiting process

LINK Link pointer in list of runnable processes

CLKT Countdown for alarm(2) measured in seconds

-r Make the execution of pstat repeat at a rate defined by the next parameter.

-t Print the table for terminals with these headings:

LOC Core location of this table entry

RAW Number of characters in raw input queue

CAN Number of characters in canonicalized input queue

OUT Number of characters in output queue

PROC Core location of the proc routine

IFLAG Input modes (see termio(7))

OFLAG Output modes (see termio(7))

CFLAG Control modes (see termio(7))

LFLAG Line discipline modes (see termio(7))

STATE Internal state:

TIMEOUT	00000001	Delay timeout in progress.
WOPEN	00000002	Waiting for open to complete.
ISOPEN	00000004	Device is open.
TBLOCK	00000010	
CARR_ON	00000020	Software copy of carrier-present.
BUSY	00000040	Output in progress.
OASLP	00000100	Wake-up when output done.
IASLP	00000200	Wake-up when input done.
TTSTOP	00000400	Output stopped by CONTROL-S.
EXTPROC	00001000	External processing.
TACT	00002000	
CLESC	00004000	Last char escape.
RTO	00010000	
TTIOW	00020000	
TTXON	00040000	
TTXOFF	00100000	
TS_RCOLL	00200000	Collision in read select.
TS_WCOLL	00400000	Collision in write select.
TS_NBIO	01000000	Tty in non-blocking mode.
TS_ASYNC	02000000	Tty in async I/O mode.
TS_STOP	04000000	Block background output.

PGRP Process group for which this is controlling terminal

LN Line discipline

DEL Number of delimiters (newlines) in canonicalized input queue

COL Calculated column position of terminal

ROW Calculated row position of terminal

IX Index to the table of core locations

-u Print information about a user process. The next argument is its address as given by ps(1). The process must be in main memory, or the file used can be a core image and the address 0.

-v Cause a number of the other flag options to give a more verbose output. Often this means that they list table entries that are not currently active or in use.

EXAMPLE

```
pstat -i
```

displays all the active inodes in a table format with headings.

FILES

/bin/pstat	
/unix	Namelist
/dev/kmem	Default source of tables

SEE ALSO

ps(1), stat(2), fs(4).

A/UX Local System Administration.

See transcript(1M)

NAME

pwck, grpck — password/group file checkers

SYNOPSIS

/etc/pwck [*file*]
/etc/grpck [*file*]

DESCRIPTION

pwck scans the password file and notes any inconsistencies. The checks include validation of the number of fields, login name, user ID, group ID, and whether the login directory and optional program name exist. The default password file is /etc/passwd.

grpck verifies all entries in the group file. This verification includes a check of the number of fields, group name, group ID, and whether all login names appear in the password file. The default group file is /etc/group.

EXAMPLES

```
pwck
```

lists inconsistencies in /etc/passwd.

```
grpck
```

lists inconsistencies in /etc/group.

FILES

/etc/pwck
/etc/group
/etc/passwd

SEE ALSO

group(4), passwd(4).

DIAGNOSTICS

Group entries in /etc/group with no login names are flagged.

See brc(1M)

See dump.bsd(1M)

NAME

reboot — reboot the operating system

SYNOPSIS

/etc/reboot [-h] [-l] [-n] [-q]

DESCRIPTION

By default, reboot causes the disks to terminate processes (using the sync command) and filesystems to be unmounted. A system restart is then initiated. Only the super-user may reboot a machine.

FLAG OPTIONS

The flag options to reboot are:

-h Halt the system and do not restart the processor.

-n Avoids the sync. It can be used if a disk is on fire.

-q Reboots quickly and ungracefully, without shutting down running processes first.

-l Disable system logging.

reboot normally logs the reboot using syslog(1M) and places a shutdown record in the login accounting file /usr/adm/wtmp. These actions are inhibited if the -n or -q options are present.

SEE ALSO

shutdown(1M), syslogd(1M), reboot(2), launch(8).

NAME

reject — prevent LP requests

SYNOPSIS

/usr/lib/reject [-r [*reason*]] [*destination...*]

DESCRIPTION

reject prevents lp(1) from accepting requests for the named *destinations*. Use lpstat(1) to find the status of *destinations*.

destination	either a printer or a class of printers.
-r[*reason*]	Associates a *reason* for preventing lp from accepting requests. This *reason* applies to all printers mentioned up to the next -r option. lp reports the *reason* when users direct requests to the named *destinations* or use lpstat(1). If you don't use the -r option or if you give -r without a *reason*, then a default *reason* is used.

FILES

/usr/lib/reject
/usr/spool/lp/*

SEE ALSO

enable(1), lp(1), lpstat(1), accept(1M), lpadmin(1M), lpsched(1M).

NAME

remshd — remote shell server

SYNOPSIS

/etc/in.remshd *host.port*

DESCRIPTION

remshd is the server for the rcmd(3N) routine and, consequently, for remsh(1N). remshd is started by inetd, see inetd(1M). The server provides remote execution facilities with authentication based on privileged port numbers.

remshd listens for service requests at the port indicated in the cmd service specification; see services(4N). When remshd receives a service request, it initiates the following protocol:

1. remshd checks the client's source port. If the port is not in the range 0-1023, it aborts the connection. The client's host address (in hex) and port number (in decimal) are the arguments passed to remshd.
2. remshd reads characters from the socket up to a null ("\0") byte. It interprets the resultant string as an ASCII number, base 10.
3. If remshd receives a port number (in step 1) which is nonzero, it interprets it as the port number of a secondary stream to use for the stderr. It then creates a second connection to the specified port on the client's machine. The source port of this second connection is also in the range 0-1023.
4. remshd checks the client's source address. If the address is associated with a host which has no corresponding entry in the host name data base (see hosts(4N)), remshd aborts the connection.
5. remshd retrieves a null-terminated user name up to 16 characters long on the initial socket. It interprets this user name as a user identity to use on the server's machine.
6. remshd retrieves a null-terminated user name up to 16 characters long on the initial socket. It interprets this user name as the user identity on the client's machine.
7. remshd retrieves a null-terminated command on the initial socket to passed to a shell. The length of the command is limited by the size of the system's argument list.

8. remshd validates the user according to the following steps. It looks up the remote user name in the password file and performs a chdir to the user's home directory. If either the lookup or chdir fail, it terminates the connection. If the user is not the superuser, (user ID 0), it consults the file /etc/hosts.equiv for a list of "equivalent" hosts. If the client's host name is in this file, the authentication is considered successful. If the lookup fails, or the user is the superuser, it checks the file .rhosts in the home directory of the remote user for the machine name and identity of the user on the client's machine. If this lookup fails, it terminates the connection.
9. remshd returns a null byte on the connection associated with the stderr and passes the command line to the normal login shell of the user. The shell inherits the network connections established by remshd.

DIAGNOSTICS

remshd returns all diagnostic messages on the connection associated with the stderr, after which it closes any network connections. It indicates an error by a leading byte with a value of 1 (it returns 0 in step 9 above if it has successfully completed all the steps up to command execution).

locuser too long
The name of the user on the client's machine is longer than 16 characters.

remuser too long
The name of the user on the remote machine is longer than 16 characters.

command too long
The command line passed exceeds the size of the argument list (as configured into the system).

Hostname for your address unknown.
There is no entry in the host name database for the client's machine.

Login incorrect.
There is no password file entry for the user name.

No remote directory.
The chdir command to the home directory failed.

`Permission denied.`
The authentication procedure described above failed.

`Can't make pipe.`
The pipe needed for the `stderr`, wasn't created.

`Try again.`
A `fork` by the server failed.

`/bin/sh: ...`
Could not start the user's login shell.

FILES

`/etc/in.remshd`

SEE ALSO

`remsh`(1N), `inetd`(1M), `rcmd`(3N).

BUGS

The authentication procedure used here assumes the integrity of each client machine and the connecting medium. This is insecure, but is useful in an "open" environment.

There should be a facility allowing all data exchanges to be encrypted.

NAME

restore — copy files from a dump.bsd archive into an existing file system

SYNOPSIS

/etc/restore [-o] [-T*file-system-type*] *key* [*argument*]...
/etc/rrestore [-o] [-T*file-system-type*] *key* [*argument*]...

DESCRIPTION

restore and rrestore recover files from a backup created with the dump.bsd(1M) and rdump commands, respectively. rrestore allows the use of a remotely connected backup device.

The -T option indicates the file sytem type, which can be either 4.2 or 5.2. The -o option indicates that restore or rrestore should assume that the backup medium contains an SVFS backup. If neither of these options is present, the type is assumed to be 5.2.

The actions of both commands are controlled by the *key* argument, which is a string of characters containing, at most, one function letter and possibly one or more function modifiers. The function letter consists of one or more characters from the set irRstx. An especially useful function letter is i, which requests an interactive restore session. The modifier consists of one or more characters from the set bfhmsvy.

Each *argument* is a file or directory name specifying a file that is to be restored. Sometimes an *argument* includes a value to be associated with a certain *key*. For example, b allows the specification of a blocking-factor as one *argument*. An *argument* for a key appears before any file or directory name, and specifically identifies a file to be recovered. If there is more than one key *argument*, then the arguments must be supplied in the same order as the associated *key*.

Unless the h key is specified (as described later in this section), the appearance of a directory name refers to the files and (recursively) subdirectories of that directory.

restore can be used together with other file-system commands. For example, see mkfs(1M) and dump.bsd(1M) to help resize a file system or recreate the same file system with more or fewer inodes to increase or decrease the number of files it can support. Disk partitioning operations are probably necessary before resiz-

ing a file system For an Apple® Hard Disk SC, use Apple HD SC Setup, as described in *A/UX Local System Administration.*

The function portion of the key is specified by one of the following letters:

`r` Read and load the backup media into the current directory. This should not be done lightly; the `r` key should only be used to restore a full dump backup onto an empty file system, or to restore an incremental dump backup after a full level zero restore. Thus

```
/etc/mkfs /dev/rdsk/cnd0sy number-of-blocks
/etc/mount /dev/dsk/cnd0sy /mnt
cd /mnt
restore r
```

is a typical sequence to restore a full dump. Another invocation of `restore` may be used to overlay the contents of an incremental dump over the full dump.

When used with the `-r` option, `restore` updates the file `restoresymtab` in the root directory to accumulate information regarding the level of backups that have been recovered for each file system. It should be removed when you no longer need the tracking information, such as when you have finished recovering from a full backup and all of its associated incremental backups.

`R` Cause `restore` to request a particular volume of a multivolume set on which to restart a full restore (see the `r` key previously described). This allows `restore` to be interrupted and then restarted.

`x` Extract the named files from the backup. If the named file matches a directory whose contents were written onto the backup, and the `h` key is not specified, the directory is recursively extracted. The owner, modification time, and mode are restored, if possible. If no file argument is given, then the root directory is extracted, which results in the entire contents of the backup being extracted, unless the `h` key is specified.

`t` List the names of the specified files if they occur on the backup. If no file argument is given, then the root directory is listed recursively, which results in the entire content of the backup being listed, unless the `h` key is specified. Note

that the `t` key replaces the function of the old `dumpdir` program.

`i` Allow interactive restoration of files from a dump backup. After reading the directory information from the backup, `restore` provides a shell-like interface that allows the user to move around the directory tree and select files to be extracted. The available commands are given next; for those commands that require an argument, the default is the current directory.

`ls` [*arg*]
List the current or specified directory. Entries that are directories are appended with `/`. Entries that have been marked for extraction are prefixed with `*`. If the verbose key is set, the inode number of each entry is also listed.

`cd` [*arg*]
Change the current working directory to the specified argument.

`pwd`
Print the full pathname of the current working directory.

`add` [*arg*]
Add the current directory, or *arg* files if specified, to the list of files to be extracted. If *arg* includes a directory, then it and all its descendents are added to the extraction list, unless the `h` key is specified on the command line. Files that are on the extraction list are prefixed with `*` when they are listed by `ls`.

`delete` [*arg*]
The current directory, or *arg* files if specified, is deleted from the list of files to be extracted. If *arg* includes a directory, then it and all its descendents are deleted from the extraction list (unless the `h` key is specified on the command line). The most expedient way to extract most of the files from a directory is to add the directory to the extraction list and then delete those files that are not needed.

`extract`
Extract from the dump backup all the files that are on

the extraction list. `restore` asks which volume the user wishes to mount. The fastest way to extract a few files is to start with the last volume and work toward the first volume.

`setmodes`
Set the owner, modes, and times of all the directories that were added to the extraction list. Nothing is extracted from the backup. This is useful for cleaning up after a restore is prematurely aborted.

`verbose`
The sense of the `v` key is toggled. When set, the verbose key causes the `ls` command to list the inode numbers of all entries. It also causes `restore` to print out information about each file as it is extracted.

`help`
List a summary of the available commands.

`quit`
`restore` immediately exits, even if the extraction list is not empty.

The following characters may be used in addition to the letter that selects the function desired.

`b` Use the associated *argument* as the blocking-factor for the records of the backup device, rather than the default blocking-factor of 1. This option should only be used with raw versions of device files.

The letters `b`, `k`, `m`, or `f` may be used at the end of the associated argument to indicate a number of blocks, kilobytes, megabytes, or feet, respectively.

Use 8 KB as the blocking-factor for the Apple Tape Backup 40SC. To restore the contents of a backup to the current directory from the tape, substitute the SCSI ID number of the tape drive for *x* and enter

```
restore -rbf 8k /dev/rmt/tcx
```

If the `-b` option is not specified, `restore` tries to determine the block size of the backup media dynamically.

`f` Read from the associated *argument* rather than the default device file `/dev/tape`. If `/etc/rrestore` is used, the

associated *argument* should include a reference to the name of the remote system where the backup device is located. A colon separates the remote-system name from the device file, as in

```
/etc/rrestore -rbf 8k server:/dev/rmt/tc3
```

If the environmental shell variable `TAPE` is set and the `f` option is not used, the value of `TAPE` is used as the device file from which data is read. If the argument associated with `f` is `-`, `restore` reads from standard input. Thus, `dump.bsd` and `restore` can be used in a pipeline to duplicate a file system with the command

```
dump.bsd 0f - /usr | (cd /mnt; restore xf -)
```

In this example, `/usr` is a mount point for another file system.

`F` Eject the 3.5-inch disk from the floppy drive when finished.

`v` Cause `restore`, which normally does its work silently, to type the name of each file it treats, preceded by its file type.

`y` Cause `restore` not to ask whether it should abort the restore if it gets a read or write error. Then `restore` always tries to skip over the bad block(s) and continue as best it can.

`m` Cause `restore` to extract by inode numbers rather than by filename. This is useful if only a few files are being extracted, and if regenerating the complete pathname to the file is to be avoided.

`h` Cause `restore` to extract the actual directory rather than the files that it references. This prevents hierarchical restoration of complete subtrees from the backup.

`s` Cause the associated `restore` argument, which is a number, to select the file on a multifile dump backup. File numbering starts at 1.

DIAGNOSTICS

Complaints about bad key characters.

Complaints if `restore` gets a read error. If `y` has been specified, or the user responds `y`, `restore` attempts to continue the restore.

If the dump extends over more than one volume, `restore` asks the user to shuttle either tape or floppy volumes around. If the `x` or `i` key is specified, `restore` also asks which volume the user wishes to mount. The fastest way to extract a few files is to start with the last volume and work toward the first volume.

There are numerous consistency checks that can be listed by `restore`. Most checks are self-explanatory or can "never happen." Common errors are given below.

filename`: not found on tape`
: The specified filename was listed in the directory of the backup, but was not found on the backup. This is caused by media read errors while looking for the file, and by recovering data from a backup created on an active file system.

`expected next file` *inumber*`, got` *inumber*
: A file that was not listed in the directory showed up. This can occur when recovering from a backup created on an active file system.

`Incremental tape too low`
: When recovering files incrementally, the backups must be copied back to primary storage in the correct order. This error can result when attempting to read from a backup that should have been read prior to the previous incremental backup.

`Incremental tape too high`
: This error can result when attempting to read a backup volume that does not begin its coverage where the previous volume left off.

`Tape read error while restoring` *filename*
: Read error occurred while skipping over an inode *inumber*.

`Tape read error while trying to resynchronize`
: A read error has occurred. If a filename is specified, then its contents are suspect. If an inode is being skipped or the backup device is trying to resynchronize, then no extracted files have been corrupted, though selected files may not be recoverable.

`resync restore, skipped` *num* `blocks`
: After a read error, `restore` may have to resynchronize itself. This message lists the number of blocks that were skipped over.

FILES

/etc/restore	
/dev/tape	The default device file
/tmp/rstdir*	File containing directories on the backup
/tmp/rstmode*	Owner, mode, and time stamps for directories
./restoresymtable	Information passed between incremental restores

SEE ALSO

dump.bsd(1M), mkfs(1M), mount(1M), newfs(1M), fstyp(2), fs(4).

BUGS

restore can get confused when doing incremental restores from dump media produced while file systems were subject to modification.

A full (level 0) dump must be done after a full restore. Because restore runs in user mode, it has no control over inode allocation; thus a full restore must be done to get a new set of directories reflecting the new inode numbering, even though the contents of the files are unchanged.

NAME

revnetgroup — reverse the netgroup file

SYNOPSIS

/etc/yp/revnetgroup [-u] [-h]

DESCRIPTION

revnetgroup reverses the netgroup file. Options are

-u reverse by username

-h reverse by hostname

Each line in the output file will begin with a key formed by concatenating the host or user name with the domain name. The key will be followed by a tab, then the comma-separated, newline-terminated list of groups to which the user or host belongs.

Exception: Groups to which everyone belongs (universal groups) will not be included in the list. The universal groups will be listed under the special name *.

NOTE

revnetgroup is a filter used in updating the /etc/yp databases. It is not expected to be of general utility.

FILES

/etc/yp/revnetgroup
/etc/netgroup

SEE ALSO

ypmake(1M), netgroup(4).

NAME

rexecd — remote execution server

SYNOPSIS

/usr/etc/in.rexecd *host.port*

DESCRIPTION

rexecd is the server for the rexec(3N) routine. The server provides remote execution facilities with authentication based on user names and encrypted passwords.

rexecd listens for service requests at the port indicated in the exec service specification; see services(4N). When it receives a service request, it initiates the following protocol:

1. The server reads characters from the socket up to a null ("\0") byte. It interprets the resultant string as an ASCII number, base 10.
2. If rexecd receives a number (in step 1) which is non-zero, it interprets it as the port number of a secondary stream to use for the stderr. It then creates a second connection to the specified port on the client's machine. The client's host address (in hex) and port number (in decimal) are the arguments passed to rexecd.
3. rexecd retrieves a null-terminated user name up to 16 characters long on the initial socket.
4. rexecd retrieves a null-terminated, encrypted, password up to 16 characters long on the initial socket.
5. rexecd retrieves a null-terminated command on the initial socket to pass to a shell. The command length is limited by the size of the system's argument list.
6. rexecd validates the user as is done at login time. If the user is authenticated, it changes to the user's home directory, and establishes user and group protections. If any of these steps fail, rexecd aborts the connection aborted and returns a diagnostic message.
7. rexecd returns a null byte on the connection associated with the stderr and passes the command line to the normal login shell of the user. The shell inherits the network connections established by rexecd.

DIAGNOSTICS

`rexecd` returns all diagnostic messages on the connection associated with the `stderr`, after which it closes any network connections. It indicates an error by a leading byte with a value of 1 (it returns 0 in step 7 above if it has successfully completed all the steps up to command execution).

`username too long`
The name is longer than 16 characters.

`password too long`
The password is longer than 16 characters.

`command too long`
The command line passed exceeds the size of the argument list (as configured into the system).

`Login incorrect.`
There is no password file entry for the user name.

`Password incorrect.`
You supplied the wrong password.

`No remote directory.`
The `chdir` command to the home directory failed.

`Try again.`
A `fork` by the server failed.

`/bin/sh: . . .`
Could not start the user's login shell.

FILES

`/usr/etc/in.rexecd`

BUGS

Indicating `Login incorrect` instead of `Password incorrect` is a security breach which allows people to probe a system for users with null passwords.

There should be a facility allowing all data exchanges to be encrypted.

NAME

rlogind — remote login server

SYNOPSIS

/etc/in.rlogind *host.port*

DESCRIPTION

rlogind is the server for the rlogin(1N) program. The server provides a remote login facility with authentication based on privileged port numbers.

rlogind listens for service requests at the port indicated in the login service specification; see services(4N). When rlogind receives a service request, it initiates the following protocol.

1. The server checks the client's source port. If the port is not in the range 0–1023, the server aborts the connection. The client's host address (in hex) and port number (in decimal) are the arguments passed to rlogind.
2. The server checks the client's source address. If the address is associated with a host that has no corresponding entry in the host name database (see hosts(4N)), the server aborts the connection.

Once it has checked the source port and address, rlogind allocates a pseudo terminal (see pty(7)), and manipulates file descriptors so that the slave half of the pseudo terminal becomes the stdin, stdout, and stderr for a login process. The login process is an instance of the login(1) program, invoked with the -r flag option. The login process then proceeds with authentication, as described in remshd(1M). If automatic authentication fails, it reprompts the user to login, as on a standard terminal line.

The parent of the login process manipulates the master side of the pseudo terminal, operating as an intermediary between the login process and the client instance of the rlogin program. In normal operation, the packet protocol described in pty(7) is invoked to provide CONTROL-S/CONTROL-D type facilities and propagate interrupt signals to the remote programs. The login process propagates the client terminal's baud rate and terminal type, as found in the environment variable, TERM; (see environ(5)).

DIAGNOSTICS

`rlogind` returns all diagnostic messages on the connection associated with the `stderr`, after which it closes any network connections. It indicates an error by a leading byte with a value of 1.

```
Hostname for your address unknown.
```

There is no entry in the host name database for the client's machine.

```
Try again.
```

A `fork` by the server failed.

```
/bin/sh: ...
```

Could not start the user's login shell.

FILES

`/etc/in.rlogind`

BUGS

The authentication procedure used here assumes the integrity of each client machine and the connecting medium. This is insecure, but is useful in an "open" environment.

NAME

route — manually manipulate the routing tables

SYNOPSIS

/etc/route [-f] [-n] [*command* [*net|host*] *destination* *gateway* [*metric*]

DESCRIPTION

route is a program used to manually manipulate the network routing tables. It normally is not needed, as the system routing table management daemon, routed(1M), should tend to this task.

route accepts two commands: add, to add a route, and delete, to delete a route.

destination is the destination host or network, *gateway* is the next-hop gateway to which packets should be addressed, and *metric* is a count indicating the number of hops to the *destination*. The metric is required for add commands; it must be zero if the destination is on a directly-attached network, and nonzero if the route utilizes one or more gateways. If adding a route with metric 0, the gateway given is the address of this host on the common network, indicating the interface to be used for transmission. Routes to a particular host are distinguished from those to a network by interpreting the Internet address associated with *destination*. The optional keywords net and host force the destination to be interpreted as a network or a host, respectively. Otherwise, if the *destination* has a "local address part" of INADDR_ANY, or if the *destination* is the symbolic name of a network, then the route is assumed to be to a network; otherwise, it is presumed to be a route to a host. If the route is to a destination connected via a gateway, the *metric* should be greater than 0. All symbolic names specified for a *destination* or *gateway* are looked up first as a host name using gethostbyname(3N). If this lookup fails, getnetbyname is then used to interpret the name as that of a network.

route uses a raw socket and the SIOCADDRT and SIOCDELRT ioctl's to do its work. As such, only the superuser may modify the routing tables.

If the -f flag option is specified, route will "flush" the routing tables of all gateway entries. If this is used in conjunction with one of the commands described above, the tables are flushed prior to the command's application.

The -n flag option prevents attempts to print host and network names symbolically when reporting actions.

DIAGNOSTICS

`add [ host | network ] %s: gateway %s flags %x`
The specified route is being added to the tables. The values printed are from the routing table entry supplied in the `ioctl` system call. If the gateway address used was not the primary address of the gateway (the first one returned by `gethostbyname`), the gateway address is printed numerically as well as symbolically.

`delete [ host | network ] %s: gateway %s flags %x`
As above, but when deleting an entry.

`%s %s done`
When the `-f` flag option is specified, each routing table entry deleted is indicated with a message of this form.

`Network is unreachable`
An attempt to add a route failed because the gateway listed was not on a directly-connected network. The next-hop gateway must be given.

`not in table`
A delete operation was attempted for an entry which wasn't present in the tables.

`routing table overflow`
An add operation was attempted, but the system was low on resources and was unable to allocate memory to create the new entry.

SEE ALSO

`routed`(1M), `intro`(5).

NAME

routed — network routing daemon

SYNOPSIS

/etc/in.routed [-d] [-g] [-s] [-q] [-t] [*logfile*]

DESCRIPTION

routed is invoked at boot time to manage the network routing tables. The routing daemon uses a variant of the Xerox NS Routing Information Protocol in maintaining up to date kernel routing table entries. It used a generalized protocol capable of use with multiple address types, but is currently used only for Internet routing within a cluster of networks.

In normal operation routed listens on the udp(5P) socket for the route service (see services(4N)) for routing information packets. If the host is an internetwork router, it periodically supplies copies of its routing tables to any directly connected hosts and networks.

When routed is started, it uses the SIOCGIFCONF ioctl to find those directly connected interfaces configured into the system and marked "up" (the software loopback interface is ignored). If multiple interfaces are present, it is assumed that the host will forward packets between networks. routed then transmits a *request* packet on each interface (using a broadcast packet if the interface supports it) and enters a loop, listening for *request* and *response* packets from other hosts.

When a *request* packet is received, routed formulates a reply based on the information maintained in its internal tables. The *response* packet generated contains a list of known routes, each marked with a "hop count" metric (a count of 16, or greater, is considered "infinite"). The metric associated with each route returned provides a metric *relative to the sender*.

response packets received by routed are used to update the routing tables if one of the following conditions is satisfied:

(1) No routing table entry exists for the destination network or host, and the metric indicates the destination is "reachable" (i.e. the hop count is not infinite).

(2) The source host of the packet is the same as the router in the existing routing table entry. That is, updated information is being received from the very internetwork router through which packets for the destination are being routed.

(3) The existing entry in the routing table has not been updated for some time (defined to be 90 seconds) and the route is at least as cost effective as the current route.

(4) The new route describes a shorter route to the destination than the one currently stored in the routing tables; the metric of the new route is compared against the one stored in the table to decide this.

When an update is applied, `routed` records the change in its internal tables and updates the kernel routing table. The change is reflected in the next *response* packet sent.

In addition to processing incoming packets, `routed` also periodically checks the routing table entries. If an entry has not been updated for 3 minutes, the entry's metric is set to infinity and marked for deletion. Deletions are delayed an additional 60 seconds to insure the invalidation is propagated throughout the local internet.

Hosts acting as internetwork routers gratuitously supply their routing tables every 30 seconds to all directly connected hosts and networks. The response is sent to the broadcast address on nets capable of that function, to the destination address on point-to-point links, and to the router's own address on other networks. The normal routing tables are bypassed when sending gratuitous responses. The reception of responses on each network is used to determine that the network and interface are functioning correctly. If no response is received on an interface, another route may be chosen to route around the interface, or the route may be dropped if no alternative is available.

`routed` supports several options:

`-d` Enable additional debugging information to be logged, such as bad packets received.

`-g` This flag is used on internetwork routers to offer a route to the "default" destination. This is typically used on a gateway to the Internet, or on a gateway that uses another routing protocol whose routes are not reported to other local routers.

`-s` Supplying this option forces `routed` to supply routing information whether it is acting as an internetwork router or not. This is the default if multiple network interfaces are present, or if a point-to-point link is in use.

`-q` This is the opposite of the `-s` option.

`-t` If the `-t` option is specified, all packets sent or received are printed on the standard output. In addition, `routed` will not divorce itself from the controlling terminal so that interrupts from the keyboard will kill the process.

Any other argument supplied is interpreted as the name of file in which `routed`'s actions should be logged. This log contains information about any changes to the routing tables and, if not tracing all packets, a history of recent messages sent and received which are related to the changed route.

In addition to the facilities described above, `routed` supports the notion of "distant" *passive* and *active* gateways. When `routed` is started up, it reads the file `/etc/gateways` to find gateways which may not be located using only information from the `SIOGIFCONF ioctl`. Gateways specified in this manner should be marked passive if they are not expected to exchange routing information, while gateways marked active should be willing to exchange routing information (i.e. they should have a `routed` process running on the machine). Passive gateways are maintained in the routing tables forever and information regarding their existence is included in any routing information transmitted. Active gateways are treated equally to network interfaces. Routing information is distributed to the gateway and if no routing information is received for a period of the time, the associated route is deleted. External gateways are also passive, but are not placed in the kernel routing table nor are they included in routing updates. The function of external entries is to inform `routed` that another routing process will install such a route, and that alternate routes to that destination should not be installed. Such entries are only required when both routers may learn of routes to the same destination.

The `/etc/gateways` is comprised of a series of lines, each in the following format:

<net | host > name1 gateway name2 metric value < `passive` *| active | external>*

The `net` or `host` keyword indicates if the route is to a network or specific host.

name1 is the name of the destination network or host. This may be a symbolic name located in `/etc/networks` or `/etc/hosts` (or, if started after `named`(1M), known to the name server), or an Internet address specified in "dot" notation; see `inet`(3N).

name2 is the name or address of the gateway to which messages should be forwarded.

value is a metric indicating the hop count to the destination host or network.

One of the keywords `passive`, `active` or `external` indicates if the gateway should be treated as passive or active (as described above), or whether the gateway is external to the scope of the `routed` protocol.

Internetwork routers that are directly attached to the Arpanet or Milnet should use the Exterior Gateway Protocol (EGP) to gather routing information rather then using a static routing table of passive gateways. EGP is required in order to provide routes for local networks to the rest of the Internet system. Sites needing assistance with such configurations should contact the Computer Systems Research Group at Berkeley.

FILES

`/etc/in.routed`	
`/etc/gateways`	for distant gateways

SEE ALSO

`route`(1M), `udp`(5P).

BUGS

The kernel's routing tables may not correspond to those of `routed` when redirects change or add routes. The only remedy for this is to place the routing process in the kernel.

`routed` should incorporate other routing protocols, such as Xerox NS and EGP. Using separate processes for each requires configuration options to avoid redundant or competing routes.

`routed` should listen to intelligent interfaces, such as an IMP, and to error protocols, such as ICMP, to gather more information. It does not always detect unidirectional failures in network interfaces (e.g., when the output side fails).

NAME

rpcinfo — report RPC information

SYNOPSIS

rpcinfo -p [*host*]

rpcinfo -u *host program-number version-number*

rpcinfo -t *host program-number version-number*

DESCRIPTION

rpcinfo makes an RPC call to an RPC server and reports what it finds.

FLAG OPTIONS

The following flag options are interpreted by rpcinfo:

-p Probe the portmapper on *host*, and print a list of all registered RPC programs. If *host* is not specified, it defaults to the value returned by hostname(1).

-u Make an RPC call to procedure 0 of *program-number* using UDP, and report whether a response was received.

-t Make an RPC call to procedure 0 of *program-number* using TCP, and report whether a response was received.

FILES

/usr/etc/rpcinfo

SEE ALSO

portmap(1M).
A/UX Network System Administration.

NAME

rstatd — kernel statistics server

SYNOPSIS

/usr/etc/rpc.rstatd

DESCRIPTION

rstatd is a server which returns performance statistics obtained from the kernel. The rstatd daemon is normally started by /etc/inetd.

FILES

/usr/etc/rpc.rstatd

SEE ALSO

inetd(1M).

NAME

runacct — run daily accounting

SYNOPSIS

/usr/lib/acct/runacct [*mmdd* [*state*]]

DESCRIPTION

runacct is the main daily accounting shell procedure. It is normally initiated via cron(1M). runacct processes connect, fee, disk, and process accounting files. It also prepares summary files for prdaily or billing purposes.

runacct takes care not to damage active accounting files or summary files in the event of errors. It records its progress by writing descriptive diagnostic messages into active. When an error is detected, a message is written to /dev/console, mail (see mail(1)) is sent to root and adm, and runacct terminates. runacct uses a series of lock files to protect against reinvocation. 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(1M) 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 runnacct, use: `nohup runacct 2> /usr/adm/acct/nite/fd2log &`

To restart runacct, use

```
nohup runacct 0601 2>> /usr/adm/acct/nite/fd2log &
```

To restart runacct at a specific *state*, use

```
nohup runacct 0601 MERGE 2>> /usr/adm/acct/nite/fd2log &
```

FILES

```
/usr/lib/acct/runacct
/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

mail(1), acct(1M), acctcms(1M), acctcom(1), acctcon(1M), acctmerg(1M), acctprc(1M), acctsh(1M), cron(1M), fwtmp(1M), acct(2), acct(4), utmp(4).

BUGS

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.

NAME

rusersd — rusers server

SYNOPSIS

/usr/etc/rpc.rusersd

DESCRIPTION

rusersd is a server which returns information for rusers(1N). The rusers daemon is normally invoked by inetd(1M).

FILES

/usr/etc/rpc.rusersd

SEE ALSO

rusers(1N), inetd(1M).

NAME

rwall — write to all users over a network

SYNOPSIS

rwall *host1 host2 ...*
rwall -n *netgroup1 netgroup2 ...*
rwall -h *host* -n *netgroup*

DESCRIPTION

rwall reads a message from standard input until end-of-file. It then sends this message, preceded by the line

```
Broadcast Message ...
```

to all users logged in on the specified host machines.

A machine can receive such a message only if it is running the Internet daemon, inetd(1M), which will invoke rwalld if /usr/etc/rpc.rwalld is listed in the file /etc/servers.

FLAG OPTIONS

The following flag options are interpreted by rwall:

-n Send the message to the specified network groups, which are defined in netgroup(4).

-h Specify *host* for *netgroup*. If this option is used, it must precede the -n option.

FILES

/usr/etc/rwall
/etc/servers

SEE ALSO

inetd(1M), rwalld(1M), shutdown(1M), wall(1M), netgroup(4), servers(4).

NAME

rwalld — network rwall server

SYNOPSIS

/usr/etc/rpc.rwalld

DESCRIPTION

rwalld is a server that handles rwall(1) and shutdown(1M) requests. It is implemented by calling wall(1) to all the appropriate network machines. The rwalld daemon is normally started by /etc/inetd.

FILES

/usr/etc/rpc.rwalld

SEE ALSO

rwall(1), wall(1), inetd(1M), shutdown(1M).

NAME

rwhod — system status server

SYNOPSIS

/etc/in.rwhod

DESCRIPTION

rwhod is the server maintaining the database used by rwho(1) and ruptime(1). Its operation is predicated on the ability to broadcast messages on a network.

rwhod both produces and consumes status information. It periodically queries the state of the system and constructs status messages that are broadcast on a network. It also listens for the status messages of other rwhod servers, validates the messages, then records them in files in the directory /usr/spool/rwho.

The rwho server transmits and receives messages at the port indicated in the rwho service specification, (see services(4N)). The messages sent and received are of the form:

```
struct      outmp {
      char  out_line[8];/* tty name */
      char  out_name[8];/* user id */
      long  out_time;   /* time on */
};

struct      whod {
      char  wd_vers;
      char  wd_type;
      char  wd_fill[2];
      int   wd_sendtime;
      int   wd_recvtime;
      char  wd_hostname[32];
      int   wd_loadav[3];
      int   wd_boottime;
      struct      whoent {
            struct      outmp we_utmp;
            int   we_idle;
      } wd_we[1024 / sizeof (struct whoent)];
};
```

All fields are converted to network byte order before being transmitted. The load averages are calculated by the kernel and represent load averages 5, 10, and 15 minutes. The host name is returned by the gethostname(2N) system call. The array at the

end of the message contains information about the users logged in to the sending machine. This includes the contents of the utmp(4) entry for each non-idle terminal line and the time since a character was last received on the terminal line.

The rwho server discards messages it receives if they don't originate at the port of a rwho server, or if the host name specified in the message contains any unprintable ASCII characters. rwhod places valid messages it receives in files named whod.*hostname* in the directory /usr/spool/rwho. These files contain only the most recent message.

rwhod generates status messages approximately every 60 seconds. rwhod performs an nlist(3C) on /unix every 10 minutes to ensure that this file is the system image currently operating.

FILES

/etc/rwhod

SEE ALSO

rwho(1), ruptime(1), services(4N).

BUGS

rwhod should relay status information between networks. People often interpret the server dying as a machine going down.

See sadc(1M)

See sadc(1M)

NAME

sadc, sa1, sa2 — system activity report package

SYNOPSIS

/usr/lib/sa/sadc [*t n*] [*file*]

/usr/lib/sa/sa1 [*t n*]

/usr/lib/sa/sa2 [-u] [-b] [-y] [-c] [-w] [-a] [-q] [-v] [-m] [-A] [-s*time*] [-e*time*] [-i*sec*]

DESCRIPTION

System activity data can be accessed at the special request of a user (see sar(1)) or automatically, on a routine basis as described here. The operating system contains a number of counters that are incremented as various system actions occur. These include counters for CPU utilization, buffer usage, disk and tape I/O activity, TTY device activity, switching and system-call, file-access, queue activity, and counters for interprocess communications.

sadc and the shell procedures sa1 and sa2 are used to sample, save, and process this data.

sadc, the data collector, samples system data *n* times every *t* seconds and writes in binary format to *file* or to standard output. If *t* and *n* are omitted, a special record is written. This facility is used at system boot time to mark the time at which the counters restart from zero. The /etc/rc entry

```
su adm -c "/usr/lib/sa/sadc /usr/adm/sa/da`date +%d`"
```

writes the special record to the daily data file to mark the system restart.

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 only once, if omitted. The entries in crontab (see cron(1M))

```
0 * * * 0,6 /usr/lib/sa/sa1
0 8-17 * * 1-5 /usr/lib/sa/sa1 1200 3
0 18-7 * * 1-5 /usr/lib/sa/sa1
```

will produce records every 20 minutes during working hours and hourly otherwise. The shell script sa2, a variant of sar(1), writes a daily report in the file /usr/adm/sa/sar*dd*. The flag options are explained in sar(1). The crontab entry

```
5 18 * * 1-5 /usr/lib/sa/sa2 -s 8:00 -e 18:01 -i 3600 -A
```

will compile a single report at 6:05 P.M. of each working day giving an hourly summary of all activity in the interval 8:00 A.M. to 6:01 P.M.

The structure of the binary daily data file is

```
struct    sa {
   struct sysinfo si;  /* defined in
                          /usr/include/sys/sysinfo.h */
   struct minfo mi;    /* defined in
                         /usr/included/sys/sysinfo.h */
   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;      /* maximum size of inode table */
   int  mszfile;          size of file table  */
   int  mszproc;       /* maximum 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 since boot*/
   long  fileovf;      /* cumulative overflows of file
                          table since boot */
   long  procovf;      /* cumulative overflows of proc
                          table since boot */
   time_t  ts;          /* time stamp  */
   int apstate;
   long  devio[NDEVS][4];/* device unit information */

#define IO_OPS 0       /* number of I/O requests since
                          boot */
#define IO_BCNT 1      /* number of blocks transferred
                          since boot */
#define IO_ACT 2       /* cumulative time in ticks when
                          drive is active */
#define IO_RESP 3      /* cumulative I/O response time in
                          ticks since boot */
};
```

FILES

/usr/lib/sa/sadc
/usr/lib/sa/sa1
/usr/lib/sa/sa2

/usr/adm/sa/sa*dd*	daily data file
/usr/adm/sa/sar*dd*	daily report file
/tmp/sa.adrfl	address file

SEE ALSO

sag(1G), sar(1), timex(1), cron(1M).
"System Activity Package" in *A/UX Local System Administration.*

NAME

sccstorcs — build RCS file from SCCS file

SYNOPSIS

sccstorcs [-t] [-v] *sccsfiles*

DESCRIPTION

sccstorcs builds an RCS file from each SCCS file specified as an argument. The deltas and comments for each delta are preserved and installed into the new RCS file in order. Also preserved are the user access list and descriptive text, if any, from the SCCS file.

The following flags are meaningful:

-t Trace only. Prints detailed information about the SCCS file and lists the commands that would be executed to produce the RCS file. No commands are actually executed and no RCS file is made.

-v Verbose. Prints each command that is run while sccstorcs is building the RCS file.

EXAMPLES

The command line:

```
sccstorcs s.getword.c
```

creates the files getword.c and getword.c,v (which should not already exist). If those files do exist, sccstorcs will exit with an error rather than overwrite them.

WARNINGS

This reference manual entry describes a utility that Apple understands to have been released into the public domain by its author or authors. Apple has included this public domain utility for your convenience. Use it at your own discretion. Often the source code can be obtained if additional requirements are met, such as the purchase of a site license from an author or institution.

FILES

/usr/ucb/sccstorcs

SEE ALSO

ci(1), co(1), rcs(1).

Walter F. Tichy, "Design, Implementation, and Evaluation of a Revision Control System," in *Proceedings of the 6th International Conference on Software Engineering*, IEEE, Tokyo, Sept. 1982.

DIAGNOSTICS

All diagnostics are written to the standard error output. Nonzero exit status on error.

BUGS

`sccstorcs` does not preserve all SCCS options specified in the SCCS file. Most notably, it does not preserve removed deltas, MR numbers, and cutoff points.

NOTES

Ken Greer

NAME

sendmail — send mail over the Internet

SYNOPSIS

/usr/lib/sendmail [*flag...*] [*address...*]

DESCRIPTION

sendmail sends a message to one or more *addresses*, routing the message over whatever networks are necessary. sendmail does inter-network forwarding as necessary to deliver the message to the correct place.

sendmail is not intended as a user interface routine; other programs provide user-friendly front ends; sendmail is used only to deliver preformatted messages.

With no flag options, sendmail reads its standard input up to an end-of-file or a line consisting of a single dot only and sends a copy of the message found there to all of the addresses listed. It determines the network(s) to use based on the syntax and contents of the addresses.

Local addresses are looked up in a file and aliased appropriately. Aliasing can be prevented by preceding the address with a backslash. Normally the sender is not included in any alias expansions; for example, if john sends to group, and group includes john in the expansion, then the letter will not be delivered to john.

Flag options are

-ba	Goes into ARPANET mode. All input lines must end with a RETURN and all messages will be generated with a RETURN at the end. Also, the From: and Sender: fields are examined for the name of the sender.
-bd	Runs as a daemon. This requires Berkeley IPC. sendmail will fork and run in the background, listening on socket 25 for incoming SMTP connections. This is normally run from /etc/inittab.
-bi	Initialize the alias database.
-bm	Delivers mail in the usual way (default).
-bp	Prints a listing of the queue.

-bs	Uses the SMTP protocol as described in RFC821 on standard input and output. This flag implies all the operations of the -ba flag that are compatible with SMTP.
-bt	Runs in address test mode. This mode reads addresses and shows the steps in parsing; it is used for debugging configuration tables.
-bv	Verifies names only; do not try to collect or deliver a message. Verify mode is normally used for validating users or mailing lists.
-bz	Creates the configuration freeze file.
-C*file*	Uses alternate configuration file. sendmail refuses to run as root if an alternate configuration file is specified. The frozen configuration file is bypassed.
-d*X*	Sets debugging value to *X*.
-F*fullname*	Sets the full name of the sender.
-f*name*	Sets the name of the from person (that is, the sender of the mail). -f can only be used by "trusted" users (normally root, daemon, and network) or if the person you are trying to become is the same as the person you are.
-h*N*	Sets the hop count to *N*. The hop count is incremented every time the mail is processed. When it reaches a limit, the mail is returned with an error message, the victim of an aliasing loop. If not specified, Received: lines in the message are counted.
-n	Doesn't do aliasing.
-o*x value*	Sets option *x* to the specified *value*. Options are described later.
-q[*time*]	Processes saved messages in the queue at given intervals. If *time* is omitted, process the queue once. *time* is given as a tagged number, with s being seconds, m being minutes, h being hours, d being days, and w being weeks. For example, -q1h30m or -

q90m would both set the timeout to one hour thirty minutes. If *time* is specified, sendmail will run in the background. This option can be used safely with -bd.

-r*name* An alternate and obsolete form of the -f flag.

-t Reads message for recipients. To:, Cc:, and Bcc: lines will be scanned for recipient addresses. The Bcc: line will be deleted before transmission. Any addresses in the argument list will be suppressed, that is, they will *not* receive copies even if listed in the message header.

-v Goes into verbose mode. Alias expansions will be announced, and so forth.

There are also a number of processing options that may be set. Normally these will only be used by a system administrator. Options may be set either on the command line using the -o flag or in the configuration file. The options are

A*file* Uses alternate alias file.

c On mailers that are considered "expensive" to connect to, doesn't initiate immediate connection. This requires queuing.

d*x* Set the delivery mode to *x*. Delivery modes are i for interactive (synchronous) delivery, b for background (asynchronous) delivery, and q for queue only – that is, actual delivery is done the next time the queue is run.

D Try to automatically rebuild the alias database if necessary.

e*x* Set error processing to mode *x*. Valid modes are m to mail back the error message, w to "write" back the error message (or mail it back if the sender is not logged in), p to print the errors on the terminal (default), and q to throw away error messages (only exit status is returned). If the text of the message is not mailed back by modes m or w and if the sender is local to this machine, a copy of the message is appended to the file dead.letter in the sender's home directory.

F*mode* The mode to use when creating temporary files.

f Saves UNIX-style `From` lines at the front of messages.

g*N* The default group ID to use when calling mailers.

H*file* The SMTP help file.

i Doesn't take dots on a line by themselves as a message terminator.

L*n* The log level.

m Sends to "me" (the sender) even in an alias expansion.

o If set, this message may have old style headers. If not set, this message is guaranteed to have new style headers (that is, commas instead of spaces between addresses). If set, an adaptive algorithm is used that will correctly determine the header format in most cases.

Q*queuedir*
Selects the directory in which to queue messages.

r*timeout*
The timeout on reads; if none is set, `sendmail` will wait forever for a mailer. This option violates the word (if not the intent) of the SMTP specification, so the timeout should probably be fairly large.

S*file* Saves statistics in the named file.

s Always instantiates the queue file, even under circumstances where it is not strictly necessary. This provides safety against system crashes during delivery.

T*time* Sets the timeout on undelivered messages in the queue to the specified time. After delivery has failed (for example, because of a host being down) for this amount of time, failed messages will be returned to the sender. The default is three days.

t*stz,dtz* Sets the name of the time zone.

u*N* Sets the default user ID for mailers.

In aliases, the first character of a name may be a vertical bar to cause interpretation of the rest of the name as a command to pipe the mail to. It may be necessary to quote the name to keep `sendmail` from suppressing the blanks from between arguments. For example, a common alias is

```
msgs: "|/usr/ucb/msgs -s"
```

Aliases may also have the syntax `:include:`*filename* to ask `sendmail` to read the named file for a list of recipients. For example, an alias such as

```
poets: ":include:/usr/local/lib/poets.list"
```

would read `/usr/local/lib/poets.list` for the list of addresses making up the group.

`sendmail` returns an exit status describing what it did. The codes are defined in `<sysexits.h>`.

`EX_OK`	Successful completion on all addresses.
`EX_NOUSER`	User name not recognized.
`EX_UNAVAILABLE`	Catchall meaning necessary resources were not available.
`EX_SYNTAX`	Syntax error in address.
`EX_SOFTWARE`	Internal software error, including bad arguments.
`EX_OSERR`	Temporary operating system error, such as "`cannot fork`".
`EX_NOHOST`	Host name not recognized.
`EX_TEMPFAIL`	Message could not be sent immediately, but was queued.

FILES

`usr/lib/aliases`	raw data for alias names
`usr/lib/aliases.pag`	
`usr/lib/aliases.dir`	data base of alias names
`usr/lib/sendmail.cf`	configuration file
`usr/lib/sendmail.fc`	frozen configuration
`usr/lib/sendmail.hf`	help file
`usr/lib/sendmail.st`	collected statistics
`usr/spool/mqueue/*`	temp files

SEE ALSO

`mail`(1), `rmail`(1), `mailq`(1M), `newaliases`(1M), `aliases`(4).
A/UX Network System Administration.

NAME

setport — set a serial port

SYNOPSIS

setport -r [-s *speed*] *device-file...*
setport -o [-s *speed*] *device-file...*

DESCRIPTION

setport adds or modifies entries for serial ports in /etc/inittab. The placeholder *device-file* is the name of an existing serial port device in /dev. For a given *device-file*, setport creates an entry in /etc/inittab, if necessary, and sets the port to allow, or disallow logins as desired.

Since setport creates entries in /etc/inittab, it may be used by a device initialization routine called by /etc/autoconfig. In this case, it is important to ensure that a device node exists in /dev before running setport from an autoconfig initialization routine.

FLAG OPTIONS

setport interprets the following arguments

-r	Respawn: set the port to permit login sessions.
-o	Set the port to off to disallow login sessions.
-s *speed*	Specify the initial baud speed to be used. The default is 9600. For modems, 1200 should usually be specified.

EXAMPLE

The following command

```
setport -r -s 19200 tty0
```

enables login sessions on serial port 0 (the "modem" port), with the initial speed set to 19200 baud.

NOTES

setport supercedes an earlier program, tty_add and should be used in place of tty_add.

FILES

`/etc/setport`	
`/etc/inittab`	The initialization table

SEE ALSO

`init`(1M), `mknod`(1M), `tty_add`(1M), `inittab`(4).

NAME

settimezone — set the local time zone

SYNOPSIS

settimezone

DESCRIPTION

settimezone provides a simple, menu-based method for setting the local time zone. Changes take effect the next time you log in. You must be superuser to run this command.

The A/UX® system clock maintains Greenwich mean time (also know as Universal Coordinated Time). Application programs use time-zone information files to calculate the local time and, therefore, the system must know the local time zone. In order to obtain the correct local time, settimezone links the proper time zone information file to /etc/zoneinfo/localtime and changes the contents of the file /etc/TIMEZONE that is used to set the TZ environment variable.

FILES

/etc/settimezone
/etc/zoneinfo/localtime
/etc/zoneinfo/*
/etc/TIMEZONE

SEE ALSO

date(1).

NAME

showmount — show all remote mounts

SYNOPSIS

showmount [–a] [–d] [–e] [*host*]

DESCRIPTION

showmount lists all the clients that have remotely mounted a file system from *host*. This information is maintained by the mountd(1M) server on *host*, and is saved across crashes in the file /etc/rmtab. The default value for *host* is the value returned by hostname(1).

FLAG OPTIONS

–d List directories that have been remotely mounted by clients.

–a Print all remote mounts in the format

hostname:directory

where *hostname* is the name of the client, and *directory* is the root of the file system that has been mounted.

–e Print the list of exported file systems.

FILES

/usr/etc/showmount

SEE ALSO

mountd(1M), exports(4), rmtab(4).

BUGS

If a client crashes, its entry will not be removed from the list until it reboots and executes umount –a.

See acctsh(1M)

NAME

shutdown — terminate all processes and bring the system down to single-user mode

SYNOPSIS

/etc/shutdown

DESCRIPTION

shutdown terminates all currently running processes in an orderly and cautious manner. shutdown interacts with the operator (who is the person who invoked shutdown) and may instruct the operator to perform some specific tasks or supply certain responses before execution can resume. shutdown goes through the following steps:

1. All users logged in to the system are notified to log out of the system by a broadcasted message. The operator may display his or her own message at this time. Otherwise, the standard backup message is displayed.
2. If the operator wishes to run the backup procedure, then shutdown unmounts all file systems.
3. The superblocks of all file systems are updated before the system is brought to single-user mode. To ensure file system integrity, a sync must be done before rebooting the system. This is usually because a process started from that file system is still running.

FILES

/etc/shutdown

DIAGNOSTICS

The most common error diagnostic is device busy. This diagnostic happens when a particular file system cannot be unmounted.

SEE ALSO

mount(1M), powerdown(1M), reboot(1M), sync(1).

NAME

slattach — attach serial lines as network interfaces

SYNOPSIS

/etc/slattach *ttyname* [*baudrate*]

DESCRIPTION

slattach is used to assign a tty line to a network interface. The *ttyname* parameter is a string of the form tty*XX* or /dev/tty*XX*. The optional *baudrate* parameter is used to set the speed of the connection. If the *baudrate* parameter is not specified, the default of 9600 is used.

After executing slattach, run ifconfig(1M) to define the network source and destination addresses. To specify the network source and destination addresses, use

ifconfig *interface-name address dest_address* up

The *interface-name* parameter is the name shown by netstat(1); either sl0 or sl1 under A/UX. *address* is the address of the local end of the slip point-to-point line. *dest_address* is the address of the remote end (the slip-serverhost) of the slip point-to-point line.

Only the superuser may attach a network interface.

To detach the interface, kill the slattach process, then use

ifconfig *interface-name* down

to quit the slip connection. *interface-name* is the name that is shown by netstat(1).

EXAMPLES

```
/etc/slattach ttyh8
ifconfig sl0 daisy-slip paris-slip

/etc/slattach /dev/tty01 1200
ifconfig sl1 daisy-slip paris-slip
```

DIAGNOSTICS

slattach produces messages indicating that the specified interface does not exist, the requested address is unknown, or the user is not privileged and tried to alter the configuration of an interface.

SEE ALSO

netstat(1), ifconfig(1M), slip(1M), hosts(4), slip.config(4), slip.hosts(4), slip.user(4).

NAME

slattconf — attach and configure serial lines as network interfaces

SYNOPSIS

/etc/slattconf *ttyname baudrate address dest-address* [*options*]

DESCRIPTION

slattconf is used to assign a tty line to a network interface and to define the network source and destination addresses. The *ttyname* parameter is a string of the form tty*XX* or /dev/tty*XX*, where *XX* is the serial port number. The *baudrate* parameter is used to set the speed of the connection. *address* is the address of the local end of the slip point-to-point line. *dest-address* is the address of the remote end (the slip-server-host) of the slip point-to-point line. The optional *options* parameters are passed to ifconfig(1M), which is invoked by slattconf.

If the slattconf command is successful, the message

sl *n*

is displayed, where *n* is replaced by the interface used by the new slip interface. For A/UX machines, you should see sl0 or sl1.

slattconf must designated be set-user-ID to the superuser in order to attach and configure a slip network interface.

To quit the slip interface, kill the slattconf process. After slattconf has been killed by hanging up the dial-up line or sending a HUP signal to the slattconf process, the slip interface is automatically taken down with ifconfig. The route to the slip interface is also removed at this time. netstat(1).

EXAMPLES

```
/etc/slattconf ttyh8 9600 joe-slip paris-slip netmask 0xffffff00
/etc/slattconf /dev/tty0 2400 chris-slip paris-slip
```

In the first example above, the user is attaching the serial port referenced by ttyh8 at 9600 baud as the slip client joe-slip to the slip server paris-slip. Note that the netmask is also specified. Both the client and server must use the same netmask. In the second example, the serial port /dev/tty0 is being attached as a slip interface. Note that the baud parameter that you specify must match the baud of your original connection to the remote machine. For example, if you are

using a 2400 baud modem to connect to the remote machine, then you should specify 2400 in the `slattconf` command line.

DIAGNOSTICS

`slattconf` produces messages indicating that the specified interface does not exist, the requested address is unknown, or the user is not privileged and tried to alter an interface's configuration.

SEE ALSO

`netstat`(1), `ifconfig`(1M), `mkslipuser`(1M), `slip`(1M), `hosts`(4), `slip.config`(4), `slip.hosts`(4), `slip.user`(4).

NAME

slip — attach a dialup serial line as a network interface

SYNOPSIS

/etc/slip

DESCRIPTION

slip is used to assign a dial-up tty line to a network interface and to define the network source and destination addresses of the point-to-point link. The assignment and definition are made on the basis of the user requesting slip, the slip user names-to-host configuration file /etc/slip.hosts, and the slip configuration files /etc/slip.config and /etc/slip.user.

A user first connects to a slip server via a dial-up or hard-wired connection, such as by using cu, kermit, or tip. After a user establishes a connection, the user invokes slip on the server to attach the serial line as a network interface.

When a user invokes slip using a dialup or hard-wired terminal, the user's ID and the available slip interfaces are checked against /etc/slip.user. If the user ID is valid and there are available unused slip interfaces, the file /etc/slip.user is updated to reflect the use of a slip interface by that user.

After invoking slip, the user no longer has access to the dial-up terminal line for issuing commands to the shell. The user must return to the local machine and invoke either slattach or slattconf to bring up the local end of the slip link. After running slattconf, the user can invoke telnet, rlogin, or other network programs from the local machine.

The kernels of both the local machine and the remote machine must be configured to support the slip interface. For A/UX machines, to configure your kernel for slip, use

```
/etc/newconfig nfs slip
```

or

```
/etc/newconfig bnet slip
```

After executing the two lines above, you need to modify /etc/inittab to turn on the networking daemons. Then reboot to run the new networking kernel.

FILES

The /etc/hosts file must be appropriately configured on the client machine before slip is invoked. The /etc/slip.config, /etc/slip.hosts, /etc/slip.user, and /etc/hosts files must be appropriately configured on the slip server before slip is invoked. If your machine is a slip client machine, the only file you have to modify is /etc/hosts.

The /etc/hosts file on the client must contain the usable Internet address when the client invokes slip. The slip-server Internet address must also be included in this file. Ask the system administrator of the slip server for the Internet addresses to include in this file. A sample /etc/hosts file is:

```
0x7F.0x00.0x00.0x01 loop lo loo
128.120.254.3 hostname1 #slip server
128.120.253.1 hostname2 #slip client
```

The first line contains the loopback address; this line is always present in the /etc/hosts file. The second line in the example is the Internet address and host address of the slip-server. The third line is the client Internet address and hostname used when the client makes a slip connection.

The system administrator of the slip host must modify the /etc/slip.config, /etc/slip.hosts, and /etc/slip.user files.

A sample slip.config configuration file is

```
# slip.config configuration file
# Each line configures a serial line
#
128.120.254.3
128.120.254.3
```

Each line is a slip-server host address for each of the slip interfaces supported by the slip-server host. In the previous example, the host has two slip interfaces available for user dialup use.

An example of a slip.hosts file is

```
# dialup slip.hosts table
# maps usercodes to host addresses
#
```

```
128.120.253.1 joe
128.120.253.2 chris
128.120.253.3 mike
128.120.253.4 linda
```

The Internet address in the first field is used when the user specified in the second field invokes `slip`.

The `slip` user file `/etc/slip.user` is not human readable. You create the `/etc/slip.user` file by creating the `/etc/slip.config` file and then running `mkslipuser`. Use the command `dslipuser` to display the contents of the user file, and to report the number of `slip` users on the system and the number of available `slip` interfaces.

DIAGNOSTICS

When a `slip` command succeeds, one of two messages is printed.

`Attaching source-host-name (`*aa.bb.cc.dd*`) to domain` *domain-name* `via` *slip-server-host-name* `(`*ee.ff.gg.hh*`)`

or

`Attaching source-host-name (`*aa.bb.cc.dd*`) to network via` *slip-server-host-name* `(`*ee.ff.gg.hh*`)`

When a `slattconf` command succeeds, the following message is displayed,

`sl`*n*

The *n* is replaced by the `slip` interface used by the new `slip` connection.

Any of the following error messages indicate that the `slip` command failed. Note that the *error-string* reports in the following messages are generated by `perror`.

```
/etc/slip.user: can't seek
/etc/slip.user: can't write
ioctl TCGETA: error-string
ioctl TCSETA: error-string
ioctl LDGETU: error-string
ioctl SIOCSIFDSTADDR: error-string
ioctl SIOCSIFADDR: error-string
```

In addition, the diagnostic `Connection failure:` may be accompanied by any of the following messages.

```
Bad login name
```

```
Can't open list of valid user-host mappings
User userid is not authorized to connect to SLIP
Invalid address aa.bb.cc.dd in hosts file
Can't open SLIP user file
Unable to lock SLIP user file
Host hostname is already attached
All lines are busy. Try again later.
```

The following error message is a warning that setting the subnet mask for the `slip` point-to-point line failed. The line is brought up using a standard internet address.

`ioctl SIOCSIFNETMASK:` *error-string*

SEE ALSO

`netstat`(1), `dslipuser`(1M), `ifconfig`(1M), `mkslipuser`(1M), `hosts`(4), `slip.config`(4), `slip.hosts`(4), `slip.user`(4).

BUGS

`slip` has a fixed definition `slip`-netmask that may be defined in `/etc/hosts`. This `slip`-netmask allows you one and only one subnet mask for all `slip` hosts. This should be configurable per `slip` host.

If the `slip` netmask is defined on the `slip` server then you must specify a matching netmask in the *netmask* option of the `slattconf` command.

`slip` lines require careful handling by the router or by special hand-installed routes on the `slip`-server-host. The Internet router shipped with A/UX (`in.routed`) handles `slip` correctly. Most Internet routers do not.

NAME

spray — spray packets

SYNOPSIS

/usr/etc/spray *host* [-l *length*] [-c *count*]

DESCRIPTION

spray sends a one-way stream of packets to *host* using RPC, and then reports how many were received by *host* and what the transfer rate was. The default value of *length* is 86 bytes (the size of the RPC and UDP headers) and the default value of *count* is the number of packets required to make the total stream size 100,000 bytes. The host name can be either a name or an Internet address.

The *length* parameter is the number of bytes in the Ethernet packet that holds the RPC call message. Since the data is encoded using XDR and XDR only deals with 32 bit quantities, not all values of *length* are possible. spray will round up to the nearest possible value. When *length* is greater than 1514, then the RPC call can no longer be encapsulated in one Ethernet packet, so the *length* field no longer has a simple correspondence to Ethernet packet size.

FILES

/usr/etc/spray

SEE ALSO

sprayd(1M).

NAME

sprayd — spray server

SYNOPSIS

/usr/etc/rpc.sprayd

DESCRIPTION

sprayd is a server which returns information for spray(1). The sprayd daemon is normally invoked by inetd(1M).

FILES

/usr/etc/rpc.sprayd

SEE ALSO

spray(1M).

NAME

StartMonitor — display a progress bar during the A/UX® boot sequence

SYNOPSIS

StartMonitor

DESCRIPTION

StartMonitor displays a Macintosh® dialog box with a progress bar during the latter stages of the A/UX boot process (after A/UX startup has passed control to the kernel and the kernel has launched init. macsysinit, which is the first entry in /etc/inittab, invokes startmac with StartMonitor as the "Finder™" application (using the -f flag of startmac).

StartMonitor receives messages via a System V message queue from other processes involved in booting the system. These messages indicate the total number of boot phases, the current boot phase, what percentage of that phase has finished, the ID of messages to be displayed, and the strings to substitute for parameters in the message strings. The message strings are stored in the string list in the StartMonitor resource file (/mac/sys/Startup System Folder/%StartMonitor). The message ID is its position in the string list.

StartMonitor exits when it receives a quit message.

Shell scripts involved in booting invoke startmsg with appropriate arguments to send messages. fsck sends messages directly to StartMonitor. After the root file system has been checked, applications that need to send messages to StartMonitor could invoke startmsg to do it for them. If many messages need to be sent, the application could create a pipe to startmsg and write startmsg argument strings into the pipe.

FILES

/mac/sys/Startup System Folder/StartMonitor
/mac/sys/Startup System Folder/%StartMonitor
/usr/include/sys/startmsg.h

SEE ALSO

macsysinit(1M), startmsg(1M).

NAME

startmsg — send messages to StartMonitor during the A/UX® boot process

SYNOPSIS

startmsg -

startmsg [-p*numphases*] [-n*nextphase*] [-d*pcntdone*] [-m*msgselector* [*substr1* ... *substr4*]] [-q]

DESCRIPTION

startmsg is used during the boot process to send messages to StartMonitor via a System V message queue. These messages control the movement of the progress bar in the StartMonitor dialog box. The messages indicate the total number of boot phases, the current boot phase, what percentage of that phase has finished, p the message to be displayed, and the substrings to be incorporated into the message. The messages are stored as an array of strings in the StartMonitor resource file, /mac/sys/Startup/System Folder/%StartMonitor. The message displayed is the one at index *msgselector* in the array.

Normally, startmsg is invoked by shell scripts such as sysinitrc. After the root file system has been checked, applications that need to send messages to StartMonitor (those that take an amount of time that could worry users) could invoke startmsg to do it for them. If many messages need to be sent, the application could create a pipe to startmsg and write startmsg argument strings into the pipe. In such cases, startmsg will consume and process one line of the input at a time, in the same order as it is generated, and exiting when an end-of-file signal arrives.

Logically, the boot process is divided into phases based on the primary activities which occur. The normal boot process consists of six phases, as is shown. Because StartMonitor does not run before phase two, the very first startmsg command issued uses the -n option to start the progress bar at phase three. Because this is approximately half way through the total number of phases, the progress bar appears at about mid-position when you first see it. Relative to the first three phases, the final three phases are time consuming, so the -d option is used, allowing incremental updating of the progress bar while you wait. Note that although you may specify a number outside the range of 1 to 100 for

pcntdone, the progress bar can not be advanced beyond the finish point for a given phase; when within the third phase of the six-phase A/UX boot process, the progress bar can not be advanced beyond the point 3/6ths of the total length of the progress bar (nor reduced to a point less than 2/6ths of the total length of the progress bar). To advance the progress bar further you must use the `-n` option and its *nextphase* argument.

Phase	Description
1.	The A/UX Startup application loads the kernel.
2.	The kernel is launched. Then, `init` spawns `macsysinit`. If the `-v` (verbose) flag option was not passed to `launch` during startup, `macsysinit` launches `macsysinitrc` which launches `StartMonitor` and `CommandShell`.
3.	The root file system is checked.
4.	`autoconfig` runs, and the device-driver startup scripts are executed.
5.	File systems other than the root file system are checked.
6.	Background processes (daemons) listed in `/etc/inittab` are spawned.

`StartMonitor` monitors the boot process after it is spawned in phase two, after which the current phase, and completion percentages within the current phase, can be established with `startmsg`.

FLAG OPTIONS

The following flag options are interpreted by `startmsg`:

`-` Read argument strings from standard input. These strings can contain any of the other `startmsg` options, just as they would be passed on the command line. This option is mutually exclusive of all others on the command line.

`-p`*numphases*
Specify the total number of phases in the boot process. This message should normally only be sent once.

`-n`*nextphase*
Specify which phase of booting is starting.

`-d`*pcntdone*
Specify what percentage of the current phase has completed.

-m*msgselector* [*substr1 ... substr4*]
Specify the index, *msgselector*, into an array of message strings to select the one to be displayed in the dialog box. Up to four substrings may also be specified. When specified, these substrings are incorporated into the the selected message string in corresponding order. So the first substring replaces the first substring placeholder in the selected message string that is stored in the `StartMonitor` resource file. The second, third, and fourth substrings specified are handled similarly.

`-q`
Send the quit signal to `StartMonitor`.

FILES

`/etc/startmsg`
`/usr/include/sys/startmsg.h`

SEE ALSO

`macsysinit(1M)`, `StartMonitor(1M)`.

startup(1M)

See acctsh(1M)

NAME

startup — run startup programs at boot time

SYNOPSIS

/etc/startup

DESCRIPTION

startup is a shell script, called from /etc/sysinitrc, which runs a set of startup routines to initialize autoconfigured modules that are part of the kernel. An example startup script is /etc/startup.d/BNET which initializes the loop interface lo0.

NOTE

startup is called from /etc/sysinitrc at system startup. It is not expected to be of general utility.

FILES

/etc/startup
/etc/startup.d/*

SEE ALSO

autoconfig(1M), sysinitrc(1M).

NAME

statd — provide crash and recovery for network locking services

SYNOPSIS

/etc/rpc.statd

DESCRIPTION

statd is a network status monitor daemon that is an intermediate version of the status monitor. It interacts with lockd(1M) to provide the crash and recovery functions for the locking services on the network file system (NFS).

/etc/statmon/current and /etc/statmon/backup are directories generated by statd. Each entry in /etc/statmon/current represents the name of the machine to be monitored by the statd daemon. Each entry in /etc/statmon/backup represents the name of the machine to be notified by the statd daemon upon its recovery.

/etc/statmon/state is generated by statd to record its version number. This version is incremented each time a crash or recovery takes place.

FILES

/etc/sm/*
/etc/sm.bak/*
/etc/state
/etc/statmon/current
/etc/statmon/backup
/etc/statmon/state

SEE ALSO

lockd(1M), sm(4).

BUGS

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

NAME

stdhosts — convert Internet addresses to standard form

SYNOPSIS

/etc/yp/stdhosts *file*

DESCRIPTION

stdhosts converts Internet addresses to a standard form. Addresses are read from a file (usually /etc/hosts).

NOTES

stdhosts is a filter used in updating the /etc/yp data bases. It is not expected to be of general utility.

FILES

/etc/yp/stdhosts	
/etc/hosts	the host table

SEE ALSO

ypmake(1M), hosts(4).

NAME

swap — add or delete disk blocks to or from the swap area

SYNOPSIS

/etc/swap -a [*swapdev* [*swaplow* [*swaplen*]]]

/etc/swap -d *swapdev* [*swaplow*]

/etc/swap -l

DESCRIPTION

swap provides a method of adding, deleting, and monitoring the system swap areas used by the memory manager.

FLAG OPTIONS

swap interprets the following flag options:

-a Add the specified swap area. If no swap area is specified, add all entries in /etc/fstab with type swap. The placeholder *swapdev* is the name of a block special device, for example, /dev/dsk/c0d0s0. The value of *swaplow* is the offset in 512-byte blocks into the device where the swap area should begin. The value of *swaplen* is the length of the swap area in 512-byte blocks. The default value for *swaplow* and *swaplen* is 0. If *swaplen* is 0, then the length of the swap area is determined from the size of the partition. See dpme(4). This option can only be used by the superuser. Swap areas are normally added by the system startup routine /etc/rc when going into multiuser mode.

-d Delete the specified swap area. The placeholder *swapdev* is the name of a block special device, for example, /dev/dsk/c0d0s0. The value of *swaplow* is the offset in 512-byte blocks into the device where the swap area should begin. The default value for *swaplow* is 0. Using this option marks the swap area as "being deleted." The system does not allocate any new blocks from the area and tries to free swap blocks from it. The area remains in use until all blocks from it are freed. This option can be used only by the superuser.

-l List the status of all the swap areas. The output has four columns:

DEV The *swapdev* device file for the swap area if one can be found in the /dev/dsk or /dev directories, and its major and minor device number, in decimal.

LOW The value of *swaplow* for the area in 512-byte blocks.

LEN The value of *swaplen* for the area in 512-byte blocks.

FREE The number of free 512-byte blocks in the area. If the swap area is being deleted, this column will be marked `(indel)`.

WARNINGS

No check is done to see if a swap area being added overlaps with an existing swap area or file system.

FILES

`/etc/fstab`
`/etc/swap`

See brc(1M)

NAME

talkd — remote user communication server

SYNOPSIS

/etc/talkd

DESCRIPTION

talkd is the server that notifies a user that somebody else wants to initiate a conversation. It acts a repository of invitations, responding to requests by clients wishing to hold a conversation.

In normal operation, a client, the caller, initiates a session by sending a CTL_MSG to the server of type LOOK_UP (see <protocols/talkd.h>). This causes the server to search its invitation tables to check if an invitation currently exists for the caller (to speak to the callee specified in the message). If the look-up fails, the caller then sends an ANNOUNCE message causing the server to broadcast an announcement on the callee's login ports requesting contact. When the callee responds, the local server uses the recorded invitation to respond with the appropriate rendezvous address and the caller and callee client programs establish a stream connection through which the conversation takes place.

FILES

/etc/talkd

SEE ALSO

talk(1N), write(1).

See init(1M)

NAME

telnetd — DARPA TELNET protocol server

SYNOPSIS

/usr/etc/in.telnetd

DESCRIPTION

telnetd is a server which supports the DARPA standard TELNET virtual terminal protocol. telnetd is invoked by the internet server (see inetd(1M)), normally for requests to connect to the TELNET port as indicated by the /etc/services file (see services(4N)).

telnetd operates by allocating a pseudo-terminal device (see pty(7)) for a client, then creating a login process which has the slave side of the pseudo-terminal as stdin, stdout, and stderr. telnetd manipulates the master side of the pseudo-terminal, implementing the TELNET protocol and passing characters between the remote client and the login process.

When a TELNET session is started up, telnetd sends TELNET options to the client side indicating a willingness to do remote echo of characters, to suppress go ahead, and to receive terminal type information from the remote client. If the remote client is willing, the remote terminal type is propagated in the environment of the created login process. The pseudo-terminal allocated to the client is configured to operate in "cooked" mode, and with XTABS and CRMOD enabled (see tty(4)).

telnetd is willing to do: echo, binary, suppress go ahead, and timing mark. telnetd is willing to have the remote client do: binary, terminal type, and suppress go ahead.

FILES

/usr/etc/in.telnetd

SEE ALSO

telnet(1C), inetd(1M), services(4N), pty(7).

BUGS

Some TELNET commands are only partially implemented.

The TELNET protocol allows for the exchange of the number of lines and columns on the user's terminal, but telnetd doesn't make use of them.

Because of bugs in the original 4.2 BSD `telnet`(1C), `telnetd` performs some dubious protocol exchanges to try to discover if the remote client is, in fact, a 4.2 BSD `telnet`(1C).

Binary mode has no common interpretation except between similar operating systems (UNIX in this case).

The terminal type name received from the remote client is converted to lowercase.

The *packet* interface to the pseudo-terminal (see `pty`(7)) should be used for more intelligent flushing of input and output queues.

`telnetd` never sends TELNET go ahead commands.

NAME

tftpd — DARPA Trivial File Transfer Protocol server

SYNOPSIS

/usr/etc/in.tftpd

DESCRIPTION

tftpd is a server supporting the DARPA Trivial File Transfer Protocol. The TFTP server operates at the port indicated in the tftp service description; see services(4N).

Since you do not have to have an account or password on the remote system to use tftp, tftpd will only allow you to access publicly readable files. This includes all users on all hosts who can be reached through the network. This may not be appropriate on all systems, and you should consider the implications before enabling tftp service.

BUGS

This server is only self-consistent. Due to the unreliability of the transport protocol (UDP) and the scarcity of TFTP implementations, it is uncertain whether it really works.

tftpd does not check the search permissions of the directories leading to the accessed files.

FILES

/usr/etc/in.tftpd

SEE ALSO

tftp(1), socket(2N), services(4N).

NAME

tic — terminfo compiler

SYNOPSIS

tic [-v[*n*]] *file* ...

DESCRIPTION

tic translates terminfo files from the source format into the compiled format. The results are placed in the directory /usr/lib/terminfo.

The -v (verbose) flag option causes tic to output trace information showing its progress. If the optional integer is appended, the level of verbosity can be increased.

tic compiles all terminfo descriptions in the given files. When a use= field is discovered, tic searches first the current file, then the master file, which is ./terminfo.src.

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

Some limitations: total compiled entries cannot exceed 4096 bytes. The name field cannot exceed 128 bytes.

FILES

/usr/bin/tic
/usr/lib/terminfo/*/*

SEE ALSO

curses(3X), terminfo(4).

BUGS

Instead of searching ./terminfo.src, tic should check for an existing compiled entry.

NAME

psbanner, pscomm, psinterface, psrv, pstext — TRANSCRIPT spooler filters for POSTSCRIPT printers

SYNOPSIS

```
/usr/lib/ps/psbanner
/usr/lib/ps/pscomm
/usr/lib/ps/psinterface
/usr/lib/ps/psrv
/usr/lib/ps/pstext
```

DESCRIPTION

These are the low-level TRANSCRIPT interface filters for use by the System V line printer spooling system. The `psinterface` shell script is a printer interface program that may be supplied to `lpadmin`. The options are as specified in the *A/UX Local System Administration.* This shell script sources a printer-specific shell script named `transcript/`*printer*`.opt` below the current working directory (the `lp` spooling directory) which may do additional printer-specific processing (for example, specify no page reversal). `psinterface` is responsible for the complete processing of the print job. If job banner break pages are enabled for this printer (and requested for this job), `psinterface` will invoke `psbanner` to format a banner break page. `psinterface` also distinguishes between text files (which get formatted) and POSTSCRIPT print files. If the input to `psinterface` does not begin with the POSTSCRIPT "magic number" (the first two characters being `%!`), `psinterface` will invoke `pstext` to create a listing of the file. If the first bytes of the input file are `%!PS-Adobe-`, and if the printer options so specify, `psinterface` will also page-reverse the file (with the `psrv` filter) before printing. `psinterface` currently recognizes three TRANSCRIPT-specific spooling options (presented to `lp` with the `-o` flag option): the `h` flag option suppresses the printing of a banner break page, the `r` flag option suppresses page reversal, and the `m` flag option causes `psinterface` to send any stream output from the execution of the user's POSTSCRIPT print file back to the user as mail.

The program `pscomm` is the lowest level filter. It manages communication with the printer, error handling, status reporting, etc. `psinterface` and `pscomm` manage a printer log file named `transcript/`*printer*`-log` (under the `lp` spooling directory). This file contains a log of each job processed, as well as any error

output from the printer. In particular, it contains messages regarding paper-out, paper-jams, and so forth. Doing a `tail` on this file will help determine the printer's status.

FILES

`/usr/lib/ps/*`	POSTSCRIPT library, prologues, filters, etc.
`/transcript/`*printer*`-log`	Printer log file.
`/transcript/`*printer*`.opt`	Printer-specific options script.
`/usr/tmpX/b*`	Break page temporary generated by `psbanner` and `psinterface`.
`/usr/tmpX/o*`	Job output temporary for mail.
`/usr/tmpX/t*`	Temporary file to format text files.

SEE ALSO

`lp`(1), `lpstat`(1), `psdit`(1), `psroff`(1), `lpadmin`(1M), `lpsched`(1M).
A/UX Local System Administration.

NAME

trpt — transliterate protocol trace

SYNOPSIS

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

DESCRIPTION

trpt prints a readable description of TCP trace records created when a socket is marked for *debugging* (see getsockopt(2N)). When you don't supply an flag option, trpt prints all the trace records grouped according to TCP connection protocol control block (PCB). The following flag options alter this:

-a in addition to the normal output, print the values of the source and destination addresses for each packet recorded

-j list only the protocol control block addresses for which there are trace records

-p*hex-address* list only trace records associated with the *hex-address* protocol control block

-s in addition to the normal output, print a detailed description of the packet-sequencing information

-t in addition to the normal output, print the values for all timers at each point in the trace

system used for debugging a system other than the default

core used for debugging a core file other than the default

We recommend the following use of trpt. Isolate the problem and enable debugging on the socket(s) involved in the connection. Find the address of the protocol control blocks associated with the sockets using netstat -A (see netstat(1N)). Then, run trpt -p and supply the associated protocol control block addresses. If there are many sockets using the debugging flag option, you might want to use the -j flag option to check for any trace records for the socket in question.

FILES

/usr/etc/trpt
/unix
/dev/kmem

SEE ALSO

getsockopt(2N), netstat(1N).

DIAGNOSTICS

no namelist

the system image doesn't contain the proper symbols to find the trace buffer.

Other diagnostics are self explanatory.

BUGS

Should also print the data for each input or output, but this is not saved in the trace record.

NAME

tty_add, tty_kill — modify the /etc/inittab file

SYNOPSIS

tty_add [-r] [-g*speed*] *device-file-name...*

tty_kill

DESCRIPTION

tty_add and tty_kill are programs designed to be run from a device-initialization routine from /etc/autoconfig. tty_add adds getty entries to /etc/inittab for the devices listed and may take two flag options (see "FLAG OPTIONS" below). tty_kill removes getty entries from /etc/inittab for which no corresponding entry in /dev exists. It would normally be run after running dev_kill(1M).

FLAG OPTIONS

tty_add interprets the following flag options:

-r Set the inittab entry to respawn; without this flag, the effective setting is off .

-g*speed*
Set *speed* that is an argument of getty (the name of an entry in the /etc/gettydefs file) to *speed*. The default value is 9600.

FILES

/etc/tty_add
/etc/tty_kill
/etc/inittab
/etc/gettydefs

SEE ALSO

autoconfig(1M), dev_kill(1M), getty(1M), setport(1M).

NOTES

The functionality of tty_add has been superceded by setport(1M). Use of setport is recommended over tty_add.

See tty_add(1M)

NAME

tunefs — tune an unmounted Berkeley 4.2 file system (UFS)

SYNOPSIS

/etc/tunefs [-p] [-m*minfree*] [-d*rotdelay*] [-e*maxbpg*] [-a*maxcontig*] [-o*optimization*] *special*

DESCRIPTION

tunefs either prints the value of or changes the dynamic parameters of a UFS file system that affects the layout policies.

FLAG OPTIONS

-p Prints the current values of the maximum number of contiguous blocks, rotation delay, blocks per cylinder group, minimum free space, and optimization. The following options are used to change these values:

-m*minfree*

Specify the percentage of space reserved from use by normal users. This value is the minimum free-space threshold. The default value of *minfree* is 10%. This value can be set to 0 for the file system, although up to a factor of three in throughput is lost over the performance obtained at a 10% threshold. Note that if the value is raised above the current usage level, users will be unable to allocate files until enough files have been deleted to get under the higher threshold.

-d*rotdelay*

Specify the expected time (in milliseconds) to service a transfer-completion interrupt and initiate a new transfer on the same disk. The option is used to decide how much rotational spacing to place between successive blocks in a file. The value of *rotdelay* must be greater than or equal to 0 and less than or equal to 16. This value is set to 4 for the A/UX release distribution of the root file system.

-e*maxbpg*

Specify the maximum number of blocks any single file can allocate out of a cylinder group before it is forced to begin allocating blocks from another cylinder group. Typically the value of *maxbpg* is set to about one quarter of the total blocks in a cylinder group. The intent is to prevent any single file from using up all the blocks in a single cylinder group, thus degrading access times for all files subsequently allocated in that cylinder group. The effect of this limit is to cause big files to do long seeks more frequently than if they were al-

lowed to allocate all the blocks in a cylinder group before seeking elsewhere. For file systems with exclusively large files, this parameter should be set higher. This value is set to 256 for the A/UX release distribution of the root file system.

−a*maxcontig*
Specify the maximum number of contiguous blocks that are laid out before forcing a rotational delay (see −d below). The default value is 1 because most device drivers require an interrupt per disk transfer. The value of *maxcontig* must be greater than 0 and less than or equal to 200. This value is set to 1 for the A/UX release distribution of the root file system.

−o*optimization*
Specify that the file system can either try to minimize the time spent allocating blocks or can attempt to minimize the space fragmentation on the disk. If the value of *minfree* (see above) is less than 10%, the file system should optimize for space to avoid running out of full-sized blocks. For values of *minfree* greater than or equal to 10%, fragmentation is unlikely to be problematical, and the file system can be optimized for time. This value is set to time for the A/UX release distribution of the root file system.

SEE ALSO
`fs`(4), `newfs`(1M).

BUGS
This program should work on mounted and active file systems. Because the superblock is not kept in the buffer cache, the changes only take effect if the program is run on unmounted file systems. To change the root file system, the system must be rebooted after the file system is tuned.

NOTES
Device drivers that can chain several buffers together in a single transfer should set *maxcontig* to the maximum chain length.

See acctsh(1M)

NAME

tzdump — time zone dumper

SYNOPSIS

tzdump [-v] [-c *cutoffyear*] [*zonename...*]

DESCRIPTION

tzdump prints the current time in each *zonename* named on the command line.

These flag options are available:

-v For each *zonename* on the command line, print the current time, the time at the lowest possible time value, the time one day after the lowest possible time value, the times both one second before and exactly at each time at which the rules for computing local time change, the time at the highest possible time value, and the time at one day less than the highest possible time value. Each line ends with isdst=1 if the given time is Daylight Saving Time or isdst=0 otherwise.

-c *cutoffyear*
Cut off the verbose output near the start of the given year.

FILES

/etc/tzdump
/etc/zoneinfo standard zone information directory

SEE ALSO

tzic(1M), ctime(3), tzfile(4).

NAME

tzic — time zone compiler

SYNOPSIS

tzic [-v] [-d *directory*] [-l *localtime*] [-p *posixrules*] [-L *leapsecondfilename*] [-s] [*filename...*]

DESCRIPTION

tzic reads text from the file(s) named on the command line and creates the time-conversion information files specified in this input. If a *filename* is -, the standard input is read.

tzic interprets the following flag options:

-v Complain if a year that appears in a data file is outside the range of years representable by time(2) values.

-d *directory*
Create time conversion information files in the named directory rather than in the standard directory named below.

-p *timezone*
Use the rules of the given time zone when handling POSIX-format time zone environment variables. tzic acts as if the file contained a link line of the form

Link timezone posixrules

-l *localtime*
Use the given time zone as local time. tzic acts as if the file contained a link line of the form

Link timezone localtime

-L *leapsecondfilename*
Read leap-second information from the file with the given name. If this option is not used, no leap-second information appears in output files.

-s Limit time values stored in ouput files to values that are the same whether they are taken to be signed or unisgned. You can use this option to generate files compatible with the System V file system (SVFS).

Input lines are made up of fields. Fields are separated from one another by any number of space characters. Leading and trailing spaces on input lines are ignored. An unquoted number sign character (#) in the input introduces a comment which extends to the end of the line the number sign character appears on. Space char-

acters and number sign characters may be enclosed in double quotes (") if they're to be used as part of a field. Any line that is blank after comment stripping is ignored. Nonblank lines are expected to be of one of three types: rule lines, zone lines, and link lines.

A rule line has the form

```
Rule NAME FROM TO TYPE IN ON AT SAVE LETTER/S
```

An example is:

```
Rule USA 1969 1973 - Apr lastSun 2:00 1:00 D
```

The fields that make up a rule line are:

NAME The (arbitrary) name of the set of rules this rule is part of.

FROM The first year in which the rule applies. The word `minimum` (or an abbreviation) means the minimum year with a representable time value. The word `maximum` (or an abbreviation) means the maximum year with a representable time value.

TO The final year in which the rule applies. In addition to `minimum` and `maximum` (as above), the word `only` (or an abbreviation) may be used to repeat the value of the *FROM* field.

TYPE The type of year in which the rule applies. If *TYPE* is –, then the rule applies in all years between *FROM* and *TO* inclusive. If *TYPE* is `uspres`, the rule applies in United States presidential election years. If *TYPE* is `nonpres`, the rule applies in years other than U.S. presidential election years. If *TYPE* is something else, then `tzic` executes the command

```
yearistype year type
```

to check the type of a year. An exit status of 0 is taken to mean that the year is of the given type; an exit status of 1 is taken to mean that the year is not of the given type.

IN The month in which the rule takes effect. Month names may be abbreviated.

ON The day on which the rule takes effect. Recognized forms include:

`5`	The fifth of the month
`lastSun`	The last Sunday in the month
`lastMon`	The last Monday in the month
`Sun>=8`	The first Sunday on or after the 8th
`Sun<=25`	The last Sunday on or before the 25th

Names of days of the week may be abbreviated or spelled out in full. Note that there must be no spaces within the *ON* field.

AT The time of day at which the rule takes effect. Recognized forms include:

`2`	Time in hours
`2:00`	Time in hours and minutes
`15:00`	24-hour time format (times after noon)
`1:28:14`	Time in hours, minutes, and seconds

Any of these forms may be followed by the letter `w` if the given time is local wall-clock time or `s` if the given time is local standard time. In the absence of `w` or `s`, wall-clock time is assumed.

SAVE The amount of time to be added to local standard time when the rule is in effect. This field has the same format as the *AT* field (although, the `w` and `s` suffixes are not used).

LETTER/S The "variable part" (for example, the `S` or `D` in `EST` or `EDT`) of time-zone abbreviations to be used when this rule is in effect. If this field is `-`, the variable part is null.

A zone line has the form

`Zone` *NAME GMTOFF RULES/SAVE FORMAT [UNTIL]*

An example is:

```
Zone Australia/South-west 9:30 Aus CST 1987 Mar 15 2:00
```

The fields that make up a zone line are:

NAME The name of the time zone. This is the name used in creating the time conversion information file

for the zone.

GMTOFF The amount of time to add to Greenwich mean time (GMT) to get standard time in this zone. This field has the same format as the *AT* and *SAVE* fields of rule lines. The field must begin with a – (minus sign) if time must be subtracted from GMT.

RULES/SAVE The name of the rule(s) that apply in the time zone or, alternately, an amount of time to add to local standard time. If this field is –, then standard time always applies in the time zone.

FORMAT The format for time-zone abbreviations in this time zone. The pair of characters `%s` is used to show where the "variable part" of the time-zone abbreviation goes.

UNTIL The time at which the GMT offset or the rule(s) change for a location. It is specified as a year, a month, a day, and a time of day. If this is specified, the time-zone information is generated from the given GMT offset and rule change until the time specified.

The next line must be a "continuation" line. This line has the same form as a zone line except that the string `Zone` and the name are omitted because the continuation line places information starting at the time specified as the *UNTIL* field in the previous line in the file used by the previous line. Continuation lines may contain an *UNTIL* field, just as zone lines do, to indicate that the next line is a further continuation.

A link line has the form

```
Link LINK-FROM LINK-TO
```

An example:

```
Link    US/Eastern    EST5EDT
```

LINK-FROM
The same as the *NAME* field in some zone line.

LINK-TO
An alternate name for the same *NAME* field as above in that zone line.

Except for continuation lines, lines may appear in any order in the input.

Lines in the file that describes leap seconds have the following form:

```
Leap YEAR MONTH DAY HH:MM:SS CORR R/S
```

An example is:

```
Leap 1974 Dex 31 23:59 + S
```

The `YEAR`, `MONTH`, `DAY`, and `HH:MM:SS` fields tell when the leap second happened. The `CORR` field should be + if a second was added or - if a second was skipped. The `R/S` field should be (an abbreviation of) Stationary if the leap second time given by the other fields should be interpreted as GMT or (an abbreviation of) Rolling if the leap second time given by the other fields should be intpreted as local wall clock time.

NOTES

For areas with more than two types of local time, you may need to use local standard time in the *AT* field of the earliest transition time's rule to ensure that the earliest transition time recorded in the compiled file is correct.

FILES

`/etc/tzic`	
`/etc/zoneinfo`	Standard directory used for created files

SEE ALSO

`tzdump`(1M), `ctime`(3), `tzfile`(4).

See mount(1M)

NAME

uucico, uushell — transfer files queued by uucp or uux

SYNOPSIS

/usr/lib/uucp/uucico [-d*spooldir*] [-g*grade*] [-r*role*] [-R] [-s*system*] [-x*debug*] [-L] [-t*turnaround*]

/usr/lib/uucp/uushell

DESCRIPTION

uucico performs the actual work involved in transferring files between systems. uucp(1C) and uux(1C) merely queue requests for data transfer which uucico processes.

uushell serves as the login shell for the user uucp. uushell sets the environment variable TZ and calls uucico.

The following options are available to uucico.

-d*spooldir*
: Use *spooldir* as the spool directory. The default is /usr/spool/uucp.

-g*grade* Only send jobs of grade *grade* or higher this transfer. The grade of a job is specified when the job is queued by uucp or uux.

-r*role* *role* is either 1 or 0; it indicates whether uucico is to start up in master or slave role, respectively. 1 is used when running uucico by hand or from cron(1M). 0 is used when another system calls the local system. Slave *role* is the default.

-R Reverse roles. When used with the -r1 option, this tells the remote system to begin sending its jobs first, instead of waiting for the local machine to finish.

-s*system*
: Call only system *system*. If -s is not specified and -r1 is specified, uucico will attempt to call all systems for which there is work. If -s is specified, a call will be made even if there is no work for that system. This is useful for polling.

-x*debug* Turn on debugging at level *debug*. Level 5 is a good start when trying to find out why a call failed. Level 9 is very detailed. Level 99 is absurdly verbose. If *role* is 1 (master), output is normally written to the standard error output. If the standard error output is unavailable,

output is written to /usr/spool/uucp/AUDIT/*system*. When *role* is 0 (slave), debugging output is always written to the AUDIT file.

-L Only call "local" sites. A site is considered local if the *device-type* field in L.sys is one of LOCAL, DIR, or TCP.

-t*turnaround*
Use *turnaround* as the line turnaround time (in minutes) instead of the default 30. If *turnaround* is missing or 0, line turnaround will be disabled. After uucico has been running in slave role for *turnaround* minutes, it will attempt to run in master role by negotiating with the remote machine. In earlier versions of uucico, a transfer of many large files in one direction would hold up mail going in the other direction. With the turnaround code working, the message flow will be more bidirectional in the short term. This option only works with newer versions of uucico and is ignored by older ones.

If uucico receives a SIGFPE (see kill(1)), it will toggle the debugging on or off.

uucico is commonly used either of two ways: as a daemon run periodically by cron(1M) to call out to remote systems, and as a "shell" for remote systems who call in. For calling out periodically, a typical line in a crontab file would be:

```
0 * * * * /usr/lib/uucp/uucico -r1
```

This will run uucico every hour in master role. For each system that has transfer requests queued, uucico calls the system, logs in, and executes the transfers. The file L.sys is consulted for information about how to log in, while L-devices specifies available lines and modems for calling.

For remote systems to dial in, an entry in the passwd(4) file must be created, with a login shell of uushell. For example:

```
nuucp:password:5:5::/usr/spool/uucppublic:/usr/lib/uucp/uushell
```

The UID for UUCP remote logins is not critical, so long as it differs from the UUCP Administrative login. The latter owns the UUCP files, and assigning this UID to a remote login would be an extreme security hazard.

FILES

/usr/spool/uucp/D.hostnameX/	
/usr/lib/uucp/	UUCP internal files/utilities
/usr/lib/uucp/L-devices	Local device descriptions
/usr/lib/uucp/L-dialcodes	Phone numbers and prefixes
/usr/lib/uucp/L.cmds	Remote command permissions list
/usr/lib/uucp/L.sys	Host connection specifications
/usr/lib/uucp/USERFILE	Remote directory tree permissions list
/usr/spool/uucp/	Spool directory
/usr/spool/uucp/AUDIT/*	Debugging audit trails
/usr/spool/uucp/C./	Control files directory
/usr/spool/uucp/D./	Incoming data file directory
/usr/spool/uucp/D.hostname/	Outgoing data file directory
/usr/spool/uucp/D.hostnameX/	Outgoing execution file directory
/usr/spool/uucp/CORRUPT/	Place for corrupted C. and D. files
/usr/spool/uucp/ERRLOG	UUCP internal error log
/usr/spool/uucp/LOGFILE	UUCP system activity log
/usr/spool/uucp/LCK/LCK..*	Device lock files
/usr/spool/uucp/SYSLOG	File transfer statistics log
/usr/spool/uucp/STST/*	System status files
/usr/spool/uucp/TM./	File transfer temp directory

/usr/spool/uucp/X./	Incoming execution file directory
/usr/spool/uucppublic	Public access directory

SEE ALSO

uucp(1C), uuq(1C), uux(1C), uuclean(1M), uuxqt(1M).

NAME

uuclean — clean up the uucp spool directory

SYNOPSIS

/usr/lib/uucp/uuclean [-d*directory*] [-m*file*] [-n*time*] [-p[*pre*]] [-s*sys*] [-w*file*]

DESCRIPTION

uuclean scans the spool directory (by default, /usr/spool/uucp) for files with the specified prefix and deletes those that are older than the specified number of hours. By default, uuclean deletes files beginning with LCK, C, X, T, TM, D, STST, and LTMP. uuclean creates a record of deletions in the spool directory called LOGDEL.

This program is typically started by cron(1M).

FLAG OPTIONS

The following flag options are interpreted by uuclean:

-d*directory* Clean *directory* instead of the spool directory.

-m*file* Send mail to the owner of the file when it is deleted. If *file* is specified, an entry is placed in *file*.

-n*time* Delete files whose age is more than *time* (hours) if the prefix test is satisfied. The default time is 72 hours.

-p*pre* Scan for files that begin with *pre*. Up to 25 -p arguments may be specified. A -p without any *pre* following causes all files older than the specified time to be deleted.

-s*sys* Examine only files destined for system *sys*. Up to 10 -s arguments may be specified.

-w*file* Find those files older than *time* (hours); however, the files are not deleted. If the argument *file* is present, the warning is placed in *file*; otherwise, the warnings go to the standard output. The default action for uuclean is to remove files that are older than a specified time (see the -n flag option).

EXAMPLES

```
uuclean -pT -pRC -n0 -m
```

removes all files in /usr/spool/uucp with a prefix of T or RC and mails notifications to the owners of the removed files.

FILES

`/usr/lib/uucp/uuclean`	
`/usr/lib/uucp`	Directory with commands used by `uuclean` internally
`/usr/spool/uucp`	Spool directory
`/usr/spool/LOGDEL`	Record of deletions

SEE ALSO

`cron`(1M), `uucp`(1C), `uux`(1C).

See uucico(1M)

NAME

uusub — monitor UUCP network

SYNOPSIS

/usr/bin/uusub [-a*sys*] [-c*sys*] [-d*sys*] [-f] [-l] [-r] [-u*hr*]

DESCRIPTION

uusub defines a uucp subnetwork and monitors the connection and traffic among the members of the subnetwork. The following flag options are available:

-a*sys* Add *sys* to the subnetwork.

-d*sys* Delete *sys* from the subnetwork.

-l Report the statistics on connections.

-r Report the statistics on traffic amount.

-f Flush the connection statistics.

-u*hr* Gather the traffic statistics over the past *hr* hours.

-c*sys* Exercise the connection to the system *sys*. If *sys* is specified as all, then exercise the connection to all the systems in the subnetwork.

The meanings of the connections report are:

sysname #call #ok time #dev #login #nack #other

where *sysname* is the name of the remote system, *#call* is the number of times the local system tries to call *sys* since the last flush was done, *#ok* is the number of successful connections, *time* is the latest successful connect time, *#dev* is the number of unsuccessful connections because of no available device (for example, ACU), *#login* is the number of unsuccessful connections because of login failure, *#nack* is the number of unsuccessful connections because of no response (for example, line busy, system down), and *#other* is the number of unsuccessful connections because of other reasons.

The meanings of the traffic statistics are:

sysname sfile sbyte rfile rbyte

where *sysname* is the name of the remote system, *sfile* is the number of files sent and *sbyte* is the number of bytes sent over the period of time indicated in the latest uusub command with the -u*hr* flag option. Similarly, *rfile* and *rbyte* are the numbers of files

and bytes received.

EXAMPLES

```
uusub -c all -u 24
```

is typically started by cron(1M) once a day.

FILES

/usr/bin/uusub	
/usr/spool/uucp/SYSLOG	system log file
/usr/lib/uucp/L_sub	connection statistics
/lib/uucp/R_sub	traffic statistics

SEE ALSO

uucp(1C), uustat(1C).

NAME

uuxqt — UUCP execution file interpreter

SYNOPSIS

/usr/lib/uucp/uuxqt [-x*debug*]

DESCRIPTION

uuxqt interprets "execution files" created on a remote system via uux(1C) and transferred to the local system via uucico(1M). When a user uses uux to request remote command execution, it is uuxqt that actually executes the command. Normally, uuxqt is forked from uucico to process queued execution files; for debugging, it may also be run manually by the UUCP administrator.

uuxqt runs in its own subdirectory, /usr/spool/uucp/.XQTDIR. It copies intermediate files to this directory when necessary.

FILES

/usr/spool/uucp/LCK.XQT	
/usr/lib/uucp/L.cmds	Remote command permissions list
/usr/lib/uucp/USERFILE	Remote directory tree permissions list
/usr/spool/uucp/LOGFILE	UUCP system activity log
/usr/spool/uucp/LCK.XQT	uuxqt lock file
/usr/spool/uucp/X./	Incoming execution file directory
/usr/spool/uucp/.XQTDIR	uuxqt running directory

SEE ALSO

uucp(1C), uux(1C), uucico(1M).

NAME

vipw — edit the password file

SYNOPSIS

vipw

DESCRIPTION

vipw edits the password file while setting the appropriate locks, and does any necessary processing after the password file is unlocked. If the password file is already being edited, then you will be told to try again later. The vi editor will be used unless the environment variable EDITOR indicates an alternate editor. vipw performs a number of consistency checks on the password entry for root, and will not allow a password file with a "mangled" root entry to be installed.

FILES

/etc/vipw

/etc/ptmp temporary lock file for editing

SEE ALSO

passwd(1), passwd(4).

NAME

volcopy, labelit — copy file systems with label checking

SYNOPSIS

/etc/volcopy [-a] [-bpi*density*] [-buf] [-feet*size*] [-reel*num*] [-s] *fsname special1 volname1 special2 volname2*

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

DESCRIPTION

volcopy makes a literal copy of the file system using a blocksize matched to the device. Flag options are

- -a Invokes a verification sequence requiring a positive operator response instead of the standard 10-second delay before the copy is made,
- -s Invokes the DEL if wrong verification sequence. (default)

Other flag options are used only with tapes:

- -bpi*density* Bits-per-inch (that is, 800/1600/6250).
- -feet*size* Size of reel in feet (that is, 1200/2400).
- -reel*num* Beginning reel number for a restarted copy.
- -buf Use double buffered I/O.

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 file system 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 (for example, labelit) and return to volcopy by exiting the new shell.

The *fsname* argument represents the mounted name (for example, root, u1, and so forth) of the file system being copied.

The *special* argument should be the physical disk section or tape (for example, /dev/rdsk/c0d0s0, /dev/rmt/0m, and so forth).

The *volname* is the physical volume name (for example, pk3, t0122, and so forth) 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.

special1 and *volname1* are the device and volume from which the copy of the file system is being extracted. The arguments *special2* and *volname2* are the target device and volume.

The arguments *fsname* and *volname* are recorded in the last 12 characters of the superblock (`char fsname[6], volname[6];`).

`labelit` can be used to provide initial labels for unmounted disk or tape file systems. With the optional arguments omitted, `labelit` prints current label values. The `-n` option provides for initial labeling of new tapes only (this destroys previous contents).

EXAMPLES

```
volcopy newsys /dev/rdsk/c0d0s0 1 /dev/rdsk/c1d0s0 1
```

copies volume 1 of the file system labeled `newsys` which is mounted on `/dev/rdsk/c0d0s0` onto volume 1 of `/dev/rdsk/c1d0s0`.

```
labelit /dev/rdsk/c1d0s0 oldsys save
```

relabels the file system mounted on `/dev/rdsk/c1d0s0` with a new *fsname* of `oldsys` and a new *volname* of `save`.

FILES

```
/etc/volcopy
/etc/labelit
/etc/log/filesave.log
```

SEE ALSO

sh(1), fs(4).

BUGS

Only device names beginning with `/dev/rmt` are treated as tapes.

`labelit` will not work on a cartridge tape system, since such a tape cannot be used as a file system. As a result, `volcopy` cannot be used with cartridge tapes, since it requires the tape to be labeled by `labelit`.

NAME

wall — write to all users

SYNOPSIS

/etc/wall

DESCRIPTION

wall reads its standard input until an end-of-file. It then sends this message to all currently logged in users preceded by:

```
Broadcast Message from...
```

It is used to warn all users, typically prior to shutting down the system.

The sender must be superuser to override any protections the users may have invoked (see mesg(1)).

EXAMPLES

```
wall
```

will broadcast the standard input to all users who are not protected against receiving messages by the mesg command.

FILES

/etc/wall
/dev/tty*

SEE ALSO

mesg(1), write(1).

DIAGNOSTICS

"Cannot send to ..." when the open on a user's tty file fails.

NAME

whodo — who is doing what

SYNOPSIS

/etc/whodo

DESCRIPTION

whodo produces merged, reformatted, and dated output from the who(1) and ps(1) commands.

EXAMPLES

```
/etc/whodo
```

will return something like the following:

```
Fri Aug 21 16:13:39 PDT 1987
A/UX
console   judy   15:55
  console    9140   0:00 ps
  console    9141   0:01 remsh
  console    9142   0:01 csh
```

FILES

/etc/whodo
/etc/passwd

SEE ALSO

ps(1), who(1).

wtmpfix(1M)

See fwtmp(1M)

See ypserv(1M)

NAME

ypinit — build and install yellow pages database

SYNOPSIS

ypinit -m

ypinit -s *master-name*

DESCRIPTION

ypinit sets up a yellow pages (YP) server's database. It can be used to set up a master server or a slave server. You must be the superuser to run it. It asks a few questions, which are self-explanatory, and reports success or failure to the terminal.

It sets up a master server using the simple model in which that server is master to all maps in the data base. This is the way to bootstrap the YP system; later if you want you can change the association of maps to masters. All databases are built from scratch, either from information available to the program at runtime, or from the ASCII data base files in /etc. These files are listed below under FILES. All such files should be in their traditional form, rather than the abbreviated form used on client machines.

A YP database slave server is set up by copying an existing database from a running server. The *master_name* argument should be the host name of YP server, which is either actually the master server for all the maps, or a server the data base of which is believed to be up-to-date and stable.

FLAG OPTIONS

-m Indicates that the local host is to be the YP master.

-s Set up a slave database. *master_name* must be an existing, reachable YP server.

FILES

/etc/yp/ypinit
/etc/passwd
/etc/group
/etc/hosts
/etc/networks
/etc/services
/etc/protocols
/etc/netgroup
/etc/ethers

SEE ALSO

makedbm(1M), ypmake(1M), yppush(1M), ypserv(1M), ypxfr(1M), ypfiles(4).

NAME

ypmake — rebuild yellow pages database

SYNOPSIS

cd /etc/yp; make [*map*] [*variable...*]

DESCRIPTION

The file Makefile in /etc/yp is used by make for building the yellow pages (YP) database. With no arguments, make creates dbm databases for any YP maps that are out-of-date and then executes yppush(1M) to notify slave databases that there has been a change.

If you supply a *map* on the command line, make will update that map only. Typing make passwd will create and do a yppush of the password database (assuming it is out of date).

Likewise, make hosts and make networks will create and do a yppush of the databases derived from the host and network files, /etc/hosts and /etc/networks.

There are three special variables that can be specified on the command line.

DIR — Gives the directory of the source files; it defaults to /etc/.

NOPUSH — When non-null, inhibits doing a yppush of the new database files. The default is the null string.

DOM — Used to construct a domain other than the master's default domain. The default is the current domainname taken from domainname(1).

During the make process, files are created with the extension .time. These are used by the Makefile to determine which databases need to be rebuilt. Two programs, stdhosts and revnetgroup, are used exclusively by the Makefile to reformat certain files before database processing.

FILES

/etc/yp/Makefile
/etc/yp/passwd.time
/etc/yp/group.time
/etc/yp/hosts.time
/etc/yp networks.time
/etc/yp/protcols.time
/etc/yp/services.time

/etc/yp/revnetgroup
/etc/yp/stdhosts

SEE ALSO

make(1), makedbm(1M), ypserv(1M).

NAME

yppasswdd — server for modifying yellow pages password file

SYNOPSIS

/usr/etc/rpc.yppasswdd *file* [–m *arg1 arg2...*]

DESCRIPTION

yppasswdd is a server that handles password change requests from yppasswd(1). It changes a password entry in *file*, which is assumed to be in the format of passwd(4). An entry in *file* will only be changed if the password presented by yppasswd(1) matches the encrypted password of that entry.

If the –m flag option is given, then after *file* is modified, a make(1) will be performed in /etc/yp. Any arguments following the flag will be passed to make.

This server is not run by default, nor can it be started up from inetd(1M). If it is desired to enable remote password updating for the yellow pages, then an entry for yppasswdd should be put in the /etc/inittab file of the host serving as the master for the yellow pages passwd file.

EXAMPLES

If the yellow pages password file is stored as /etc/yp/src/passwd, then to have password changes propagated immediately, the server should be invoked as

```
/usr/etc/rpc.yppasswdd /etc/yp/src/passwd –m \
passwd DIR=/etc/yp/src
```

Note: The above command can be entered on one line by omitting the backslash.

FILES

/usr/etc/rpc.yppasswdd
/etc/yp/Makefile

SEE ALSO

yppasswd(1), ypmake(1M), ypwhich(1M), passwd(4), yp-files(4),
A/UX Network System Administration.

CAVEAT

This server will eventually be replaced with a more general service for modifying any map in the yellow pages

NAME

yppoll — what version of a YP map is at a YP server host

SYNOPSIS

yppoll [-h *host*] [-d *domain*] *mapname*

DESCRIPTION

yppoll asks a ypserv process what the order number is, and which host is the master YP server for the named map. If the server is a v.1 YP protocol server, yppoll uses the older protocol to communciate with it. In this case, it also uses the older diagnostic messages in case of failure.

FLAG OPTIONS

-h *host*

Ask the ypserv process at *host* about the map parameters. If *host* isn't specified, the YP server for the local host is used. That is, the default host is the one returned by ypwhich(1M). *host* may be specified either as a name or an Internet address of the form *ww.xx.yy.zz*.

-d *domain*

Use *domain* instead of the default domain.

FILES

/etc/yp/yppoll

SEE ALSO

ypserv(1M), ypfiles(4).
A/UX Network Applications Programming.

NAME

yppush — force propagation of a changed YP map

SYNOPSIS

yppush [-d *domain*] [-v] *mapname*

DESCRIPTION

yppush copies a new version of a Yellow Pages (YP) map from the master YP server to the slave YP servers. It is normally run only on the master YP server by the Makefile in /etc/yp after the master databases are changed. It first constructs a list of YP server hosts by reading the YP map ypsrvs within the *domain*. Keys within the map ypsrvs are the ASCII names of the machines on which the YP servers run.

A "transfer map" request is sent to the YP server at each host, along with the information needed by the transfer agent (the program which actually moves the map) to call back the yppush. When the attempt has completed (successfully or not), and the transfer agent has sent yppush a status message, the results may be printed to standard ouput. Messages are also printed when a transfer is not possible; for instance when the request message is undeliverable, or when the timeout period on responses has expired.

Refer to ypfiles(4) and ypserv(1M) for an overview of the yellow pages.

FLAG OPTIONS

-d Specify a *domain*.

-v Verbose. This causes messages to be printed when each server is called, and for each response. If this flag is omitted, only error messages are printed.

FILES

/etc/yppush
/etc/yp/*domainname*/ypsrvrs.*dir*
/etc/yp/*domainname*/ypsrvrs.*pag*

SEE ALSO

ypserv(1M), ypxfr(1M), ypfiles(4).
AIUX Network Applications Programming.

BUGS

In the current implementation (version 2 YP protocol), the transfer agent is `ypxfr`, which is started by the `ypserv` program. If `yppush` detects that it is speaking to a version 1 YP protocol server, it uses the older protocol, sending a version 1 `YPPROC_GET` request and issues a message to that effect. Unfortunately, there is no way of knowing if or when the map transfer is performed for version 1 servers. `yppush` prints a message saying that an "old-style" message has been sent. The system administrator should later check to see that the transfer has actually taken place.

NAME

ypserv, ypbind — yellow pages server and binder processes

SYNOPSIS

/etc/ypserv
/etc/ypbind

DESCRIPTION

The yellow pages (YP) provides a simple network lookup service consisting of databases and processes. The databases are dbm(3) files in a directory tree rooted at /etc/yp. These files are described in ypfiles(4). The processes are /etc/ypserv, the YP database lookup server, and /etc/ypbind, the YP binder. The programmatic interface to YP is described in ypclnt(3N). Administrative tools are described in yppush(1M), ypxfr(1M), yppoll(1M), ypwhich(1M), and ypset(1M). Tools to see the contents of YP maps are described in ypcat(1M), and ypmatch(1). Database generation and maintenance tools are described in ypinit(1M), ypmake(1M), and makedbm(1M).

Both ypserv and ypbind are daemon processes typically activated at system startup time from /etc/inittab. ypserv runs only on YP server machines with a complete YP database. ypbind runs on all machines using YP services, that is, both YP servers and clients.

The ypserv daemon's primary function is to look up information in its local database of YP maps. The operations performed by ypserv are defined for the implementor by the "YP protocol specification," and for the programmer by the header file <rpcsvc/yp_prot.h>. Communication to and from ypserv is by means of RPC calls. Lookup functions are described in ypclnt(3N), and are supplied as C-callable functions in /lib/libc. There are four lookup functions, all of which are performed on a specified map within some YP domain: Match, Get_first, Get_next, and Get_all. The Match operation takes a key, and returns the associated value. The Get_first operation returns the first key-value pair from the map, and Get_next can be used to enumerate the remainder. Get_all ships the entire map to the requester as the response to a single RPC request.

Two other functions supply information about the map, rather than map entries: `Get_order_number`, and `Get_master_name`. In fact, both order number and master name exist in the map as key-value pairs, but the server will not return either through the normal lookup functions. (If you examine the map with `makedbm`(1M), however, they will be visible.)

Other functions are used within the YP subsystem itself, and are not of general interest to YP clients. They include `Do_you_serve_this_domain?`, `Transfer_map`, and `Reinitialize_internal_state`.

The function of `ypbind` is to remember information that allows client processes on a single node communicate with some `ypserv` process. `ypbind` must run on every machine which has YP client processes; `ypserv` may or may not be running on the same node, but must be running somewhere on the network.

The information `ypbind` remembers is called a *binding*: the association of a domain name with the internet address of the YP server, and the port on that host at which the `ypserv` process is listening for service requests. The process of binding is driven by client requests. As a request for an unbound domain is received, the `ypbind` process broadcasts on the net, trying to find a `ypserv` process that serves maps within that domain. Since the binding is established by broadcasting, there must be at least one `ypserv` process on every net. Once a domain is bound by a particular `ypbind`, that same binding is given to every client process on the node. The `ypbind` process on the local or on a remote node may be queried for the binding of a particular domain by using the `ypwhich`(1) command.

Bindings are verified before they are given out to a client process. If `ypbind` is unable to speak to the `ypserv` process to which it's bound, it marks the domain as unbound, tells the client process that the domain is unbound, and tries to bind the domain once again. Requests received for an unbound domain will fail immediately. In general, a bound domain is marked as unbound when the node running `ypserv` crashes or gets overloaded. In such a case, `ypbind` will to bind any YP server (typically one that is less-heavily loaded) available on the net.

`ypbind` also accepts requests to set its binding for a particular domain. The request is usually generated by the YP subsystem itself. `ypset`(1M) is a command to access the `Set_domain` fa-

cility. It is for unsnarling messes, not for casual use.

Note: If the file `/etc/yp/ypserv.log` exists when `ypserv` starts up, log information will be written to this file when error conditions arise.

FILES

`/etc/ypserv`
`/etc/ypbind`

SEE ALSO

`ypcat`(1M), `ypmatch`(1M), `yppush`(1M), `ypwhich`(1M), `ypxfr`(1M), `ypset`(1M), `ypclnt`(3N), `ypfiles`(4).
A/UX Network Applications Programming.

NAME

ypset — point ypbind at a particular server

SYNOPSIS

ypset [-V1] [-h *host*] [-d *domain*] *server*

ypset [-V2] [-h *host*] [-d *domain*] *server*

DESCRIPTION

ypset tells ypbind to get YP services for the specified *domain* from the ypserv process running on *server*. If *server* is down, or isn't running ypserv, this is not discovered until a YP client process tries to get a binding for the domain. At this point, the binding set by ypset will be tested by ypbind. If the binding is invalid, ypbind will attempt to rebind for the same domain.

ypset is useful for binding a client node which is not on a broadcast net, or is on a broadcast net which isn't running a YP server host. It also is useful for debugging YP client applications, for instance, where a YP map only exists at a single YP server host.

In cases where several hosts on the local net are supplying YP services, it is possible for ypbind to rebind to another host even while you attempt to find out if the ypset operation succeeded. That is, you can type ypset host1 and then ypwhich, which replies: host2, which can be confusing. This is a function of the YP subsystem's attempt to load-balance among the available YP servers, and occurs when host1 does not respond to ypbind because it is not running ypserv (or is overloaded), and host2, running ypserv, gets the binding.

server indicates the YP server to bind to, and can be specified as a name or an IP address. If specified as a name, ypset will attempt to use YP services to resolve the name to an IP address. This will work only if the node has a current valid binding for the domain in question. In most cases, *server* should be specified as an IP address.

Refer to ypfiles(4) and ypserv(1M) for an overview of the yellow pages.

FLAG OPTIONS

-V1 Bind *server* for the (old) Version 1 YP protocol.

-V2 Bind *server* for the (current) Version 2 YP protocol.

If no version is supplied, ypset first attempts to set the domain for the (current) Version 2 protocol. If

this attempt fails, `ypset`, then attempts to set the domain for the (old) Version 1 protocol.

`-h` *host* Set `ypbind`'s binding on *host* instead of locally. The argument *host* can be specified as a name or as an IP address.

`-d` *domain* Use *domain* instead of the default domain.

FILES

`/etc/yp/ypset`

SEE ALSO

`ypserv`(1M), `ypwhich`(1M), `ypfiles`(4).
A/UX Network Applications Programming.

NAME

ypxfr — transfer a YP map from some YP server to here

SYNOPSIS

ypxfr [-f] [-h *host*] [-d *domain*] [-c] [-C *tid prot ipadd port*] *mapname*

DESCRIPTION

ypxfr moves a YP map to the local host by making use of normal YP services. It creates a temporary map in the directory /etc/yp/domainname (which must already exist), fills it by enumerating the map's entries, fetches the map parameters (master and order number) and loads them. It then deletes any old versions of the map and moves the temporary map to the real mapname.

If ypxfr is run interactively, it writes its output to the terminal. However, if it's invoked without a controlling terminal, and if the log file /etc/yp/ypxfr.log exists, it will append all its output to that file. Since ypxfr is most often run from cron(1M), or by ypserv, you can use the log file to retain a record of what was attempted, and what the results were.

For consistency between servers, ypxfr should be run periodically for every map in the YP data base. Different maps change at different rates: the services.byname map may not change for months at a time, for instance, and may therefore be checked only once a day in the wee hours. You may know that mail.aliases or hosts.byname changes several times per day. In such a case, you may want to check hourly for updates. A crontab(1) entry can be used to perform periodic updates automatically. Rather than having a separate crontab entry for each map, you can group commands to update several maps in a shell script. Examples (mnemonically named) are in /etc/yp: ypxfr_1d.sh, ypxfr_2d.sh, and ypxfr_1h.sh. They can serve as reasonable first cuts.

Refer to ypfiles(4) and ypserv(1M) for an overview of the yellow pages.

FLAG OPTIONS

-f Force the transfer to occur even if the version at the master is not more recent than the local version.

`-c` Don't send a "Clear map" request to the local `ypserv` process. Use this flag if `ypserv` is not running locally at the time you are running `ypxfr`. Otherwise, `ypxfr` will complain that it can't talk to the local `ypserv`, and the transfer will fail.

`-h` *host* Get the map from *host*, regardless of what the map says the master is. If *host* is not specified, `ypxfr` will ask the YP service for the name of the master and try to get the map from there. *host* may be a name or an internet address in the form *ww.xx.yy.zz*.

`-d` *domain* Specify a domain other than the default domain.

`-C` *tid prog ipadd port*
This option is only for use by `ypserv`. When `ypserv` invokes `ypxfr`, it specifies that `ypxfr` should call back a `yppush` process at the host with IP address *ipaddr*, registered as protocol *prot*, listening on port *port*, and waiting for a response to transaction *tid*.

FILES

```
/etc/ypxfr
/etc/yp/ypxfr.log
/etc/yp/ypxfr_1d.sh
/etc/yp/ypxfr_2d.sh
/etc/yp/ypxfr_1h.sh
```

SEE ALSO

`ypserv`(1M), `yppush`(1M), `ypfiles`(4),
A/UX Network Applications Programming.

Table of Contents

Section 7: Drivers and Interfaces for Devices

NAME

intro — introduction to device drivers and interfaces

DESCRIPTION

The entries in this section provide useful information for users and programmers, although programmers may be able to benefit more.

Users need to know what device files are typically associated with which devices or ports so that the commands that accept device files as arguments can be specified accurately. For example, /dev/tty0 and /dev/tty1 refer to serial ports, and /dev/dsk/c0d0s0 through /dev/dsk/c7d0s0 refer to slice 0 of each of the hard disks set to SCSI ID 0 through 7.

To rebuild or make custom-named device files, users need to know what major device numbers correspond to particular device drivers, as well as the operational or addressing modes selected by the minor device numbers associated with a particular device driver. Note that the A/UX autoconfiguration utilities automatically create and remove device files (with default filenames) whenever necessary for a particular device configuration they are establishing (see autoconfig(1M), newconfig(1M), and newunix(1M)).

This section also includes information helpful to readers who wish to manipulate a device directly through its corresponding interface. For example, you can format a cartridge tape through the general mt device interface to tape devices. These device-level interfaces allow access to device-specific functions, such as cartridge tape formatting, that would not normally be available as a standard operation for all tape devices. Non-universal operations typically would not be available through the flag options for the standard programs, such as tar and cpio, described in Section 1 of *A/UX Command Reference*.

Programmers who wish to write programs that can access all the peculiarities of specific devices may find the standard I/O library lacking the necessary capabilities. If you need to know this information, Section 7 describes the special system calls, or ioctls, associated with particular device drivers.

The device files identified with the letter "P" following the section number are part of the A/UX POSIX environment. The differences between the A/UX environment and the A/UX POSIX environment are described in *A/UX Guide to POSIX*, which is provided in *A/UX Programming Languages and Tools, Volume 1*.

User interface

Although a programmer knows that each device is controlled by a designated device driver, this fact is usually less well known to users. Because a disk device is selected and accessed properly whenever files are manipulated, users are sheltered from learning about the requirements of the disk device driver. However, for devices other than disk devices, there are often various addressing options or operating modes that require awareness from users. While the provisions of the file system helps users transparently access the correct disk device for read and write operations, other devices require a more visible, low-level interface. To provide a flexible way to access to these device-level and interface-level features, A/UX stretches the file-system model to encompass references to devices other than disks.

The file-system files that reference a device (or communcations port) are called *device files*. Device files are special because the data written to them does not usually reside on the disk. For example, the device file for a terminal (`/dev/console`) stays a constant size when data is written to it. In such a case, the data is not written to a disk file at all. Instead, the data is written to the actual device that the device file references.

When initially created, the device file is given two unchangeable attributes that no other files have: a major device number that selects a device driver, and a minor device number that selects a particular device out of several possible devices, or a particular operating mode from among several operating modes. The meaning of the minor device number varies from device driver to device driver. Selecting reading or writing operations for a device is usually a function of the command line within which the device filename appears: sometimes the command itself works unidirectionally (`restore`, `dump.bsd`, `finc`); other times the command is bidirectional and the direction of data flow is determined by flag options or associated arguments (`tar`, `pax`, `cpio`, `dd`).

Device files are available for use only within certain command lines. Usually, but not always, commands that accept device file as arguments have metanames such as *device-file* or *devname* within their syntax descriptions.

The device filenames are customarily derived from the hardware configuration. However, some of the device filenames are seemingly arbitrary for various historical reasons.

Sometimes a device file references a port rather than an actual device. For example, a printer connected through the printer port can be referenced as `/dev/tty1`, as described in `serial`(7). In this case, the device file references the first serial port. Conventionally, you can use the first Macintosh serial port to attach a serial printer. It is also possible to attach a modem or an A/UX user terminal to this port.

Major and minor device numbers

Programs that support devices, such as hard disks and tape drives, are known as device drivers. One device driver usually controls all the instances of one type of device. The exception is the serial device driver. It provides low-level support for a variety devices that are capable of communicating over a serial communications port. For example, a serial device driver helps control user terminals, modems, serial printers, and similar serial input/output devices.

To allow many devices of the same type to be controlled by the same device driver, each can be assigned a unique minor number using `mknod`(1M). This number is passed to the device driver for interpretation. By convention, the minor number may appear as the last part of the name of the device file for particular classes of devices. For example, `/dev/tty0` refers to the first serial port, or the port with minor device number 0. Note that this is a naming convention only, and is achieved by using `mknod`. In reality, device selection is realized through the minor number exclusively. The `-l` option of `ls` will show what major and minor device numbers have been assigned to a device file of a particular name, helping you verify whether a device file was named reasonably well.

The minor number is often used by a device driver as an inidication of the intended operational modes. For example, the device files for cartridge tape drives that contain an `n` suffix select a non-rewinding mode of tape operation through a minor number that is correctly interpreted by the device driver.

NAME

appletalk — general AppleTalk socket interface and STREAMS controls

DESCRIPTION

This manual page describes the AppleTalk I/O control calls (see ioctl(2)), device files, and the general nature of the A/UX AppleTalk interface.

Before beginning, several points should be noted. The AppleTalk library routines automatically set up and invoke the correct ioctl requests that are necessary for most AppleTalk requirements. While the ioctls give the programmer more control than the AppleTalk library routines, they require a much greater understanding of the A/UX implementation of AppleTalk. In addition, AppleTalk ioctl calls are subject to change, while AppleTalk library functions will not change. It is, therefore, strongly recommended that the library routines be used whenever possible instead of the more complicated ioctl calls.

AppleTalk Protocols

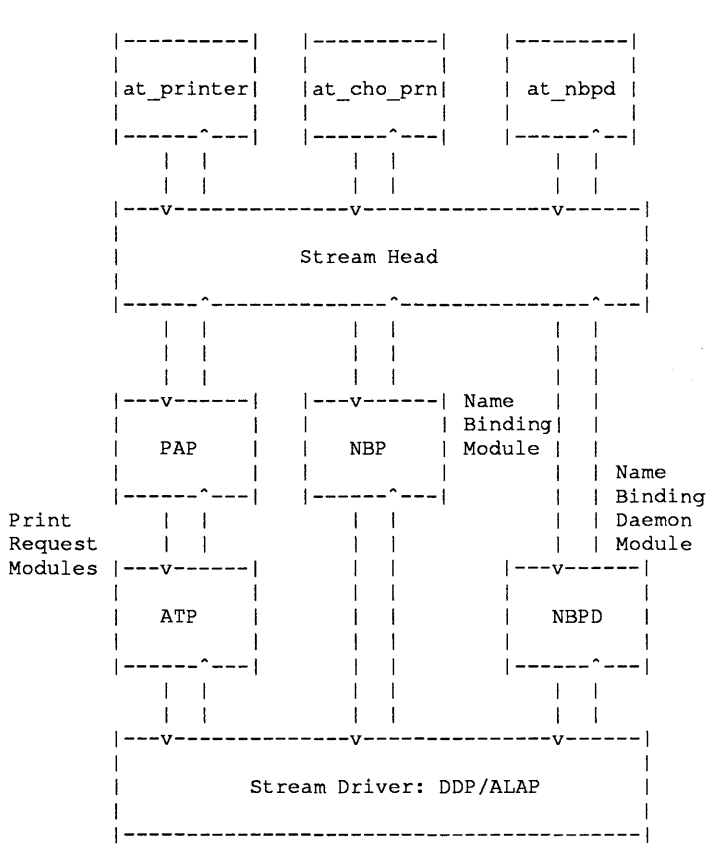
AppleTalk is implemented as a protocol stack, consisting of a set of layers, with one or more protocols per layer. This set of protocols corresponds roughly to the layers of the Open Systems Interconnection (OSI) reference model published by the International Standards Organization (ISO). Each layer is built on top of (and uses) the previous layer. The order of layers, from lowest (closest to the physical transport) to highest (closest to the application), is as follows:

Link layer	AppleTalk Link Access Protocol (ALAP)
Network layer	Datagram Delivery Protocol (DDP)
Transport layer	AppleTalk Transaction Protocol (ATP) Name Binding Protocol (NBP) Routing Table Maintenance Protocol (RTMP)
Session layer	Zone Information Protocol (ZIP) Printer Access Protocol (PAP)

The lower layers (ALAP/DDP) are normally used for new network testing and development, such as building a new layer using TCP/IP on top of DDP.

The AppleTalk Model

Most AppleTalk protocol layers are implemented as Streams modules. The two exceptions are the DDP and ALAP layers, which are implemented as Streams drivers. The majority of applications require the programmer to push one or more modules into the open stream in order to achieve the proper layering for that application. The following diagram describes the A/UX implementation of AppleTalk.



The first application illustrated (`at_printer`) shows the configuration for communicating with a network print server. Note that the ATP module must be pushed before the PAP module. While it is possible to reverse the pushing order, unpredictable

results can occur if this is done.

The second and third applications (at_cho_prn and at_nbpd) are normally used together. When AppleTalk is brought up, a special application daemon, nbpd, is invoked. It opens an AppleTalk socket and pushes the module at_nbpd into the stream. This application daemon is used by subsequent applications, such as at_nvelkup(1), to open a socket and push the module at_nbp into the stream. Modules at_nbp and at_nbpd communicate at the ALAP level to complete users' requests for name binding information.

NAME

`console` — keyboard/screen driver

DESCRIPTION

The keyboard and video screen driver provides access to the system's console keyboard and screen. Running in its default configuration, it provides emulation of an ANSI standard screen and keyboard combination. Various `ioctl` commands allow the user to configure it for use along with the mouse in a more interactive environment.

The screen driver is a streams based driver. Before use, a line discipline may have to be pushed onto the device's stream. Under most conditions, this is done automatically by the operating system. When an application opens `/dev/console` explicitly, and it is not already open, it may be necessary to push such a line discipline. There are three ways to do this.

`ioctl (`*fd*`, I_PUSH,` *line*`) ;`	Unconditionally pushes a line discipline.
`line_push (`*fd*`) ;`	Pushes a line discipline if one is not already pushed.
`/etc/line_sane` *fd*	Same as previous description, but as an application in a shell script.

In the previous examples, *fd* is a file descriptor of the open device. The last two methods are preferred, as they will only push a line discipline if required and they can be used on nonstreams drivers without adverse effects.

When `/dev/console` is opened with a line discipline pushed on it, it will respond to all the `ioctl`s and modes described in `termio`(7). Without the line discipline, it will only respond to the flags described under `c_cflag` in `termio`(7). Setting the number of bits/character to `CS8` will put the screen into reverse video when it is in terminal emulation mode.

ANSI Compatible Escape Sequences

The terminal emulator responds to the following ANSI escape sequences.

`'\b'`	backspace–move 1 column left

'\r'	carriage return–move to column 1
'\n'	linefeed–move to next line
'\t'	tab
ESC '='	turn on keypad mode
ESC '>'	turn off keypad mode
ESC 'D'	line feed
ESC 'M'	scroll down
ESC '7'	save cursor position
ESC '8'	restore cursor position
ESC '[' 's'	save cursor position
ESC '[' 'u'	restore cursor position
ESC '[' 'H'	home cursor
ESC '[' 'f'	home cursor
ESC '[' [*n*] 'A'	up *n*, if *n* = 0 go up 1
ESC '[' [*n*] 'B'	down *n*, if *n* = 0 go down 1
ESC '[' [*n*] 'D'	left *n*, if *n* = 0 go left 1
ESC '[' [*n*] 'C'	right *n*, if *n* = 0 go right 1
ESC '[' [*n*] 'J'	0->erase to end of page 1->erase to beginning of page 2->erase whole page
ESC '[' [*n*] 'K'	0->erase to end of line 1->erase from beginning of line 2->erase whole line
ESC '[' *n* 'L'	scroll down *n* lines
ESC '[' *n* 'M'	scroll up *n* lines
ESC '[' *n* '@'	insert *n* spaces
ESC '[' *n* 'P'	delete *n* characters
ESC '[' *n* 'm'	7->reverse video 0->normal video
ESC '[' *n* 'n'	6-> return cursor position as the string

	ESC '[' *row* ';' *col* 'R'
ESC '[' *n* 'Z'	tab to *n**8
ESC '[' [*n1*] ';' [*n2*] 'r'	move to column *n2*, line *n1*
ESC '[' [*n1*] ';' [*n2*] 'f'	set scrolling region from row *n1* to row *n2*
ESC '[' [*n1*] ';' [*n2*] 'H'	set scrolling region from row *n1* to row *n2*
ESC '[' '?' 'b' 'h'	enable scrolling region
ESC '[' '?' 'b' 'l'	disable scrolling region
ESC '[' '?' '5' 'h'	set to normal video (black on white)
ESC '[' '?' '5' 'l'	reverse video (white on black)

Note: Characters in single quotes (') are literal. ESC is the escape character. Other symbols represent strings of ASCII numbers which represent their decimal equivalent. [*n*] represents an optional number. If it is not present, 0 is used in its place. When sending row/column numbers to the screen, the upper left corner has address row=1, column=1.

If the keyboard is not in "keypad mode," it returns the characters on the keys. If it is in "keypad mode," then the following escape sequences are generated:

KEY LEGEND	SEQUENCE
0	ESC 'O' 'p'
1	ESC 'O' 'q'
2	ESC 'O' 'r'
3	ESC 'O' 's'
4	ESC 'O' 't'
5	ESC 'O' 'u'
6	ESC 'O' 'v'
7	ESC 'O' 'w'
8	ESC 'O' 'x'
9	ESC 'O' 'y'

.	ESC 'O' 'n'
-	ESC 'O' 'm'
ENTER	ESC 'O' 'M'

Function Keys

On the Apple® Extended keyboard, the extra keys (the function keys and the group of six keys above the arrow keys) are mapped to the following simple key sequences:

KEY LEGEND	SEQUENCE
F1	SOH '@' CR
F2	SOH 'A' CR
F3	SOH 'B' CR
F4	SOH 'C' CR
F5	SOH 'D' CR
F6	SOH 'E' CR
F7	SOH 'F' CR
F8	SOH 'G' CR
F9	SOH 'H' CR
F10	SOH 'I' CR
F11	SOH 'J' CR
F12	SOH 'K' CR
F13	SOH 'L' CR
F14	SOH 'M' CR
F15	SOH 'N' CR
DEL	SOH 'O' CR
END	SOH 'P' CR
PAGE DOWN	SOH 'Q' CR
HELP	SOH 'R' CR
HOME	SOH 'S' CR
PAGE UP	SOH 'T' CR

where SOH = 1 and CR = 13.

Note: These sequences end in a RETURN which makes the A/UX line discipline send the current input line to the current reading process. While doing this (assuming normal terminal settings), the RETURN is translated to a newline character.

The CONTROL Key

The CONTROL key sets the 6th and 5th bits of a character to 0, regardless of the key pressed. Thus, pressing CONTROL and w (octal 167) at the same time yields the same character as pressing CONTROL and W (octal 127) or CONTROL and 7 (octal 067), namely,

(octal 027).

IOCTLS

The display responds to a large number of `ioctls` which affect its behavior in many different ways. It responds to all the `ioctls` defined in `termio(7)` in the normal manner. Since it is a streams device, all other `ioctls` must be called using the streams indirect `ioctl` call function, `I_STR`. With this is passed a packet describing the `ioctl` to be executed. This packet is of type `struct strioctl` and is described in `sys/stropts.h`. It has 4 fields.

`ic_cmd`	The command (`ioctl`) requested to be executed.
`ic_dp`	A pointer to any data to be read/written to by the `ioctl` (parameters or returned results, for example).
`ic_len`	The size of the data in bytes.
`ic_timout`	A timeout (how long to wait for the `ioctl` to complete before returning an error); −1 means no timeout (or wait forever).

If `s` is a data structure of type `struct strioctl`, then an `ioctl` call is made using a call of the following form.

```
ioctl(fd, I_STR, &s)
```

Further examples of such calls are given later in this document.

Screen `ioctls` fall into three main areas: control of the keyboard, control of the video generation hardware, and control of the mouse. All the following `ioctls`, and the symbols defined with them are defined in the include file `sys/video.h`.

Keyboard `ioctls`

The keyboard operates in two modes, either as an ANSI standard keyboard generating characters in the normal manner, or in "raw" mode where each keystroke is generated (both up and down) and passed directly without any modification.

`VIDEO_RAW`	This `ioctl` puts the keyboard into "raw" mode. The key codes returned are the key codes directly from the keyboard. (Refer to the keyboard documentation for a list of these.) The most significant bit indicates whether the

keystroke is up or down. One exception is the mouse escape character (see VIDEO_MOUSE later in this section).

VIDEO_ASCII — This mode is an emulation of an ANSI standard terminal keyboard (including the keypad).

Video ioctls

The video screen can be used in two different ways, either as a terminal emulation (in which case the interface is in the normal manner), or as a bit mapped screen, mapped into a process's address space using the phys(2) system call. There is no explicit way to shift from one mode to the other; one simply stops sending characters to the terminal (including turning off ECHO) and writes to the bitmap.

VIDEO_SIZE — This ioctl returns the size (in characters) of the screen when used in terminal emulation mode. The result is two longs representing the horizontal and vertical sizes.

VIDEO_SETDEF — This sets the screen to its default, which is 1 bit per pixel mode, black on white.

VIDEO_SETCOLOR — Takes a parameter of the type video_color that describes the foreground and background colors to which the screen is set.

VIDEO_ADDR — This ioctl returns (as a longword) the physical address of the video RAM on the video board (suitable for use in a phys(2) system call).

VIDEO_REFRESH — This ioctl rewrites the entire screen, including the borders; it also clears all text from the screen and moves the cursor to the upper left corner. It takes no parameters. It is intended to be used by implementers of utilities that write directly to the screen (via phys(2), for example) when they are exiting and wish to set

the screen back to a known state. This is the only way to get the kernel to rewrite the screen's borders.

VIDEO_PIXSIZE — This ioctl returns a parameter of type struct video_size (also defined in <sys/video.h>) in which it places information about the size of the screen in pixels. The data structure contains three fields.

- pix_scr_x — The width of the visible part of the screen in pixels.
- pix_scr_y — The height of the visible part of the screen in pixels.
- pix_mem_x — The number of pixels (both visible and invisible) between the start of a line and the start of the next line.

Mouse ioctls

It is possible to attach the mouse to the keyboard device using the VIDEO_MOUSE system call. This can happen only if the mouse is not currently opened in any other way (see mouse(7)). When in mouse mode, changes in the mouse are converted into characters and inserted into the keyboard's input stream. This mode is most useful in conjunction with the keyboard mode VIDEO_RAW. The mouse input is always preceded by the character MOUSE_ESCAPE and can take one of two forms depending on whether the display is in VIDEO_M_BUTTON mode or not. If it is, then the next character is a 1 or 0 depending on the state of the mouse button (1 if it is down). In this mode, mouse information is only placed into the input stream when the state of the mouse button changes. In the other mode, the display places 2 bytes into the input stream whenever the mouse moves or the state of the button changes. The 2 bytes have the following format.

Byte 0 — Bit 7: The state of the mouse button–0 for down.

Bits 0–6: Two's complement of the mouse movement in the Y axis since the last entry.

Byte 1 — Bit 7: Always 1.

Bits 0–6: Two's complement of the mouse movement in the X axis since the last entry.

The mouse `ioctls` are

`VIDEO_MOUSE` — Put the display into the `VIDEO_MOUSE` mode as described earlier. This will fail if the mouse is already in use.

`VIDEO_NOMOUSE` — Take the display out of `VIDEO_MOUSE` mode. This is the default when the display is opened for the first time.

`VIDEO_M_BUTTON` — Put the display into a mode (described earlier) where changes only in the mouse button find their way into the keyboard input stream.

`VIDEO_M_ALL` — This option makes all mouse changes (including changes in mouse position) to be put into the keyboard's input stream. This is the default when entering `VIDEO_MOUSE` mode.

`VIDEO_M_DELTA` — This call returns the change in mouse position since it was last called (or since the display was put into `VIDEO_MOUSE` mode). It returns two shorts (horizontal displacement followed by vertical).

`VIDEO_M_ABS` — This also works only in `VIDEO_MOUSE` mode; it returns the *absolute* mouse position (relative to 0 when the system was booted). It returns two shorts (horizontal displacement followed by vertical).

EXAMPLES

The following is an example of streams `ioctl`s. It opens the keyboard and removes any line disciplines (after first saving their states); then it puts the display into `VIDEO_RAW` and `VIDEO_MOUSE` modes and reads the input, displaying it to the standard output. When a character code 1 (from the key "s") is found, it stops and puts the display back into `VIDEO_ASCII` and `VIDEO_NOMOUSE` modes. It then pushes the line discipline back on and restores its state.

```
#include <sys/stropts.h>
#include <sys/termio.h>
#include <sys/video.h>
#include <fcntl.h>
main()
{
   struct termio t;
   char c;
   short ss;
   int fd, line;
   struct strioctl s;

   fd = open("/dev/console", O_RDWR); /*open the keyboard*/
   ioctl(fd,TCGETA,&t);               /*save the old tty*/
                                      /*state*/
   line = ioctl(fd,I_POP,0);          /*remove the line*/
                                      /*discipline and*/
                                      /*remember if there*/
                                      /*was one*/
   s.ic_timout = -1;                  /*set the streams*/
                                      /*timeout to infinity*/
   s.ic_len = 0;                      /*put keyboard into*/
   s.ic_cmd = VIDEO_RAW;              /*raw mode*/
   if (ioctl(fd,I_STR,&s) < 0)
           goto quit;
   s.ic_len = 0;                      /*attach to mouse*/
   s.ic_cmd = VIDEO_MOUSE;
   if (ioctl(fd,I_STR,&s) < 0)
           goto quit;
   ioctl(fd,I_FLUSH,FLUSHRW);         /*flush input to*/
                                      /*put us in a known*/
                                      /*starting state*/
   for(;;) {                          /*loop reading input*/
           if (read(fd,&c,1) < 0)     /*and displaying it*/
               break;
           if (c == MOUSE_ESCAPE) {
                   if (read(fd,&ss,s) < 0)
                           break;
                   printf("m = 0x%04x\n",c&0xffff);
                   continue;
```

```
            }
            if (c == 1)             /*quit on char.*/
                                    /*code 1*/
                    break;
            printf("c = 0x%02x\n",c&0xff);
    }
quit:
    s.ic_len = 0;                   /*set the keyboard*/
                                    /*back to*/
    s.ic_cmd = VIDEO_NOMOUSE;       /*a sane state*/
    ioctl(fd,I_STR,&s);
    s.ic_len = 0;
    s.ic_cmd = VIDEO_ASCII;
    ioctl(fd,I_STR,&s);
    if (line == 0)                    /*if required*/
            ioctl(fd,I_PUSH,"line"); /*push a line*/
                                      /*discipline*/
    ioctl(fd,TCSETA,&t);              /*restore its modes*/
}
```

FILES

/dev/console
/dev/mouse
/usr/include/sys/video.h
/usr/include/sys/stropts.h
/usr/include/termio.h

SEE ALSO

line_sane(1M), ioctl(2), phys(2), line_push(3), mouse(7), termio(7).

NAME

error — error-logging interface

DESCRIPTION

Minor device 0 of the error driver is the interface between a process and the system's error-record collection routines. Only a single process with superuser permission may open the driver for reading. Each read retrieves an entire error record; the record is truncated if the read request is for less than the record's length.

FILES

/dev/error

SEE ALSO

errdemon(1M).

NAME

fd — 3.5-inch disk device driver

DESCRIPTION

The fd device driver provides an interface to two types of Apple® 3.5-inch disk drives. The standard drive supports a single-sided format providing 400 kilobytes (KB) of storage and a double-sided format providing 800 KB of storage. The Apple SuperDrive (formerly named the Apple FDHD™ drive) supports the 400 KB and 800 KB formats as well as industry-standard 720 KB and 1440 KB double-sided formats. The storage format, or density, is associated with individual floppy disks when they are formatted. Partitions and partition maps are not supported for floppy disks.

Three main classes of floppy access are supported: fixed-density devices, autodensity devices, and special ioctl devices.

Fixed-density Devices

These devices require that the floppy disk match a specified density. If a fixed-density device encounters the wrong density disk, the driver immediately ejects the disk, prints an informational console message, and returns EINVAL.

Autodensity Devices

These devices automatically adjust to any valid floppy disk format as long as they can be supported by the drive.

Special ioctl Devices

These devices are used only for issuing special ioctl calls (FD_GETMETER, FD_SETMETER, FD_GETTUNE, and FD_SETTUNE). During an open system call, no check is made for the presence of a floppy disk or the current status of the device, that is, the disk drive. This allows these special ioctl calls to be used at any time without interfering with other users of the device. No other I/O operations are allowed to a special ioctl device. Both the AL_EJECT and GD_PARTSIZE calls are also allowed for special ioctl devices.

The driver accepts a variety of the following ioctl functions. The first parameter for each of these functions is *file-descriptor*, which should be the raw device file corresponding to the drive, such as /dev/rdsk/c8d?s0.

`ioctl` (*file-descriptor*, `AL_EJECT, 0`)
Eject the disk. The file descriptor corresponds to the character device for the floppy device file. It may be appropriate to unmount a file system before ejecting the disk.

`ioctl` (*file-descriptor*, `GD_PARTSIZE, 0`)
Return the format of the media. The value returned is the number of blocks on the disk, as follows:

`800`	400 KB
`1600`	800 KB
`1440`	720 KB
`2880`	1440 KB
`0`	unformatted
`EI0`	empty drive

`ioctl` (*file-descriptor*, `UIOCFORMAT`, *fmt*)
`ioctl` (*file-descriptor*, `FD_FMTONLY`, *fmt*)
`ioctl` (*file-descriptor*, `FD_VFYONLY`, *fmt*)
Use `UIOCFORMAT` to format and verify the disk in one operation; use `FD_FMTONLY` or `FD_VFYONLY` to either format or verify the disk as a one operation. The third parameter to this function call contains the address of a `diskformat` structure that is defined in `/usr/include/sys/diskformat.h`. The `d_lhead` field of this structure determines the number of sides to be formatted. If it is set to 0, the disk is formatted in a one-sided, 400 KB format. If the same `d_lhead` field is nonzero, the `d_dens` field of the structure determines the density to which the disk is to be formatted. Values for `d_dens` are as follows:

`400`	400 KB
`720`	720 KB
`800`	800 KB
`1440`	1440 KB

All other fields of the `diskformat` structure are ignored by most device drivers. However, `d_fcyl` and `d_lcyl` may sometimes be used. For example, the floppy device driver uses these two quantities as the first and last cylinders to be formatted.

If both `d_lhead` and `d_dens` are set to their default values (`DISK_DEFAULT`), the density that is selected is

based on the device class. Autodensity devices, such as `/dev/rfd/d0`, default to a double-sided 800 KB format for standard media or default to a double-sided 1440 KB format for high-density media. Fixed-density devices, such as `/dev/rfd/d1m720`, create a fixed-density format only.

Conflicting density specifications, such as setting `d_dens` to `720` in an attempt to format the media in the drive referenced through `/dev/rfd/d0m1440`, causes `EINVAL` to be returned. Similarly, if the requested density is unavailable for the given device and current media, `EINVAL` is returned. Examples include referencing a non-SuperDrive drive with `/dev/rfd/d0m1440` or attempting to format a low-density disk in a 1440 KB format.

Before an ioctl function for formatting floppy disks is honored, the device must have been opened in exclusive-use mode with the `O_EXCL` flag.

`ioctl` (*file-descriptor*, `FD_GETMETER`, *meter*)
Return the current statistics counters for the *file-descriptor* device driver. The value of *meter* is the address of an `fd_meter` structure, as defined in `/usr/include/sys/fdioctl.h`.

`ioctl` (*file-descriptor*, `FD_SETMETER`, *meter*)
Copy the `fd_meter` structure at *meter* into the internal memory of the device driver. This is useful to clear the counters after doing a `FD_GETMETER` to gather statistics.

`ioctl` (*file-descriptor*, `FD_GETSTAT`, *int*)
Return the current status of the drive or media as a bit mask:

`STAT_FDHD` (0x01)
The drive is SuperDrive.

`STAT_2SIDED` (0x02)
The drive is a double-density drive.

`STAT_NODRIVE` (0x04)
No drive is present.

`STAT_NODISK` (0x08)
No 3.5-inch disk is in the drive.

`STAT_WRTENAB` (0x10)
The 3.5-inch disk in the drive is write-enabled.

STAT_1MBMEDIA (0x20)
The 3.5-inch disk in the drive is not a high-density disk.

ioctl (*file-descriptor*, FD_GETTUNE, *tune*)
Return the current settings of the tunable error thresholds in the fd device driver. The parameter *tune* is the address of a fd_tune structure as defined in the header file /usr/include/sys/fdioctl.h.

ioctl (*file-descriptor*, FD_SETTUNE, *tune*)
Copy the fd_tune structure at *tune* into the internal memory of the device driver.

FILES

Fixed-density Devices

/dev/fd/d[01]m*dens*
/dev/rfd/d[01]m*dens*

where *dens* is specified as 400, 720, 800, or 1440.

Autodensity Devices

/dev/dsk/c8d[01]s0
/dev/rdsk/c8d[01]s0
/dev/floppy[01]
/dev/rfloppy[01]
/dev/fd/d[01]
/dev/rfd/d[01]

Special ioctl Devices

/dev/rfd/d0x
/dev/rfd/d1x

Header Files

/usr/include/sys/ssioctl.h
/usr/include/sys/fdioctl.h

SEE ALSO

cpio(1), eject(1), tar(1), diskformat(1M), mkfs(1M), mount(1M), umount(1M), ioctl(2), open(2).

WARNINGS

Changing error thresholds with the *tune* parameter should not be necessary. Adjustments should be made with extreme care!

NOTES

Appending e to a device file causes the driver to eject the disk on close. Appending w causes open to block until a disk is inserted; the wait is interruptible, returning EINTR. Appending ew

does both.

High-density disks should only be formatted as 1440 KB while standard disks may only be formatted as 400 KB, 720 KB, or 800 KB. It is possible to format high-density disks as 400 KB or 800 KB on systems that do not support the SuperDrive drive. If an illegally formatted disk is encountered, the driver immediately ejects the disk, prints an informational console message, and returns `EINVAL`.

Opening in `O_EXCL` mode prevents the driver from complaining about illegally formatted disks and also makes fixed-density devices behave in autodensity mode. While this behavior is implemented to allow reformatting, it may be deliberately exploited to extract data from an improperly formatted disk.

Simultaneous access to a drive is limited to similar modes. This is enforced by only allowing users access to the drive through the same device file. This prevents incompatible combinations of density, wait-for-insert, and eject-on-close options.

The 720 KB and 1440 KB formats are not interleaved. As a result, reading or writing in small block sizes can be quite slow.

NAME

forwarder — forwarder device driver

DESCRIPTION

The forwarder is a specalized streams device driver written so as to be able to run on a wide range of front end processors (FEP).

The FEP generally has a CPU, a memory, I/O circuitry devices, and a means of communicating with the host Macintosh® II via the NuBus™. (Modules are normally downloaded onto the FEP, allowing for offloading of the host processor.)

The forwarder software is actually duplicated; identical copies are kept in the kernel on the host and in the minioperating system found on the FEP. The two copies work together (as a matched pair) to pass messages and data across the NuBus. From the kernel, the forwarder looks like a stream driver; from the actual stream driver (or modules), it looks like a stream head.

The forwarder software knows that there is a processing or space separation (the NuBus) between the operating system and the remote modules and streams driver. It is the only module that needs to know about this division of powers; it hides this fact from the other layers.

Because the NuBus exists, however, the implementor must be aware of some stream restrictions. Any operation that uses the forwarder must pass through the forwarder's queue processing. For example,

```
q->q_next->q_next
```

would be incorrect because it is trying to access the queue beyond the forwarder, and that is impossible. Careful thought and an understanding of the forwarder's task should help prevent such errors.

When it is next to a forwarder, the stream head behaves differently when it receives an I_PUSH ioctl. It first checks the module ID number downstream. If the ID number is ≥ FORWARDERMIN but ≤ FORWARDERMAX, it sends an I_PUSH via an M_IOCTL message. The forwarder passes the request to its twin on the board, which tries to open the indicated module. The forwarder then responds with an "acknowledge" if the open was completed. If the open was not completed successfully, a "negative acknowledge" is returned. If the module is not found on the board, a message is returned to that effect and the stream head continues

the push as if the forwarder were not there. The process is the same for popping, except that there is no "not found" case.

Control of the forwarder is done via stream I_STR ioctls. The following stream I_STR ioctls, defined in <fwd.h>, are available.

I_FWD_LOOKUP — Returns a table of the installed application strings and places it in the location pointed to by arg->ic_dp. An I_FWD_LOOKUP call returns a table into arg->ic_dp, where the line entries are of type struct fwd_entry, found in <fwd.h>. The length of the table is found in arg->ic_len but is always less than the stream maximum of 1024 Kbytes.

I_FWD_RESET — Resets the board into a state ready for downloading. This ioctl must be used when the system first comes up, or when an FEP panic occurs. An I_FWD_RESET call also disables any application currently talking to the board if EIO errors are detected for that application. Note that with many FEPs, the software cannot issue a reset to the board. In this case, if the forwarder has lost communication with its twin, I_FWD_RESET will have no effect, and you reboot the system to reset the forwarder.

I_FWD_DOWNLD — Causes the binary data contained in fwd_record.data to be downloaded to the FEP starting at FEP memory location fwd_record.begin. The structure fwd_record is defined in <fwd.h>.

I_FWD_UPLD — Causes the binary data to be uploaded from the FEP memory into the data field fwd_record.data. The value in fwd_record.ld_length is the number of bytes to be uploaded from the

FEP. The structure `fwd_record` is defined in `<fwd.h>` .

`I_FWD_START` Instructs the loader to transfer execution to the address contained in `fwd_entry.start`. The name field is placed in the forwarder's application table.

EXAMPLES

```
int dev_fd;
struct strioctl i_str;

        if((dev_fd = open(dev_file, O_NDELAY)) < 0)
                HANDLE_ERROR();

        i_str.ic_cmd = I_FWD_DOWNLD;
        i_str.ic_timout = 4;
        i_str.ic_len = fwd_record.begin;
        i_str.ic_dp = fwd_record;

        if(ioctl(dev_fd, I_STR, &i_str) < 0)
                HANDLE_ERROR();
```

FILES

`/dev/fwdicp11`
`/etc/startup.d/fwdicp.d/at_load`
`/etc/startup.d/fwdicp.d/tt_load`

SEE ALSO

`fwd_lkup`(1M), `fwdload`(1M).
AT&T UNIX System V STREAMS Programming Guide.

NAME

gd — generic disk interface

DESCRIPTION

The gd device driver provides a generic interface to disk devices. A variety of devices are supported, and support for new hardware may be added via autoconfiguration utilities. Consult the specific hardware manual for add-on devices to see if they use the generic disk interface.

For SCSI devices, the driver makes a distinction between disks that support the SCSI common command set and those that don't. The Apple® document *SCSI Command Protocol 062-2075* defines characteristics of the common command set. Certain features, such as changing the reporting of soft errors, may not be supported by hardware that does not implement the common command set. The SCSI driver depends on the controller's ability to save configuration information when the drive is turned off. Hardware that does not save this information requires additional device-specific software. The document *Building A/UX Device Drivers* supplies technical information on extending the generic disk driver.

Device Naming

A device controller corresponds to an A/UX® major device number. For SCSI devices, a controller corresponds to a SCSI ID. For NuBus™ based hardware, the disk controller is a single card. Each controller may have up to eight drives associated with it. For most devices, each drive would be a separate spindle with its own set of platters. For SCSI devices, each drive is a SCSI logical unit. For many SCSI devices with integral controllers, only one drive is possible.

A drive is further divided into slices (or partitions). A **slice** is a group of blocks used for a single purpose on a single drive. Most often, a slice corresponds to a file system. Slice 1 on the disk with the root file system is assumed to be a swap area at boot time. Slice 30 is assumed to contain a Macintosh® file system.

In the `/dev/dsk` and `/dev/rdsk` directories, devices are named according to their controller, disk, and slice number. `/dev/dsk/c0d0s0` would be SCSI ID 0, the first drive, and the first partition. `/dev/dsk/c9d0s0` might be assigned to NuBus slot nine. However, the low-level operating-system drivers do not access devices by name; instead, they use a pair of numbers called

the major and minor device number. A disk controller is assigned a major number by either the autoconfiguration process or by the system designers. For SCSI disks, the major numbers from 24 through 31 are reserved for SCSI disk-device IDs 0 through 7. However, these assignments are subject to change in later releases of the operating system.

A minor number is calculated from the drive and the slice number:

minor = *drive* * 32 + *slice*

There are a maximum of 256 minor numbers for each major number. There may be as many as eight drives per controller and 32 slices per drive.

Data Structures on Disk

The first block of the physical disk (block 0) is reserved by the Macintosh Operating System. Block 1 of a disk used by A/UX (each physical block is always 512 bytes) contains one or more disk partition map entries. This data structure defined in `dpme`(4) assigns areas of the disk to the various available operating systems. A/UX maintains specific operating information that further describes each partition. This data structure resides in the `dpme_boot_args` field of `dpme`. It is defined by the A/UX `bzb`(4) data structure. Information in `bzb` is used by `autorecovery`(8), the kernel, and other utilities to further define the use of a partition.

An A/UX disk partition provides three methods of compensation for flawed or bad sectors on the disk. In most cases, disk hardware or firmware remaps the bad sector without further involvement by the operating system. If the disk hardware is deficient, however, A/UX maintains a pool of spare sectors at the end of the data area of the partition. The data structures defining these spares are described in `altblk`(4). If neither of these methods is available, the `autorecovery` program may be used to assign the bad sector to an unused inode. When this alternative is used, a bad sector does not cause mischief on a mounted file system, but the bad sector continues to be present when accessed by other methods, such as an image copy via `dd`(1).

Partition Mapping

A/UX disk partitioning allows slices of a disk to be allocated to operating systems, users, or applications. A partition is a group of disk blocks that are assigned a name and a type. The utility dp(1M) is one way partitions may be created and manipulated. Although any number of partitions may reside on a drive, the number of concurrently available partitions is limited to 32. A set of device control codes have been developed to allow partitions to be selectively attached to the minor number of a given drive and then detached when their usefulness is ended.

To allow the system to be booted, three partitions are normally assigned (or associated) by default. When slice 0 is first accessed, the first partition which has the *type* field set to `Apple_UNIX_SVR2`, which has the root file system bit set in the block 0 block data structure, and whose `autorecovery`(8) cluster number matches the `autorecovery` cluster number requested by the A/UX Startup Shell booter (or matches the default cluster number of 0) is associated.

When slice 1 is first accessed, a partition is associated that matches the A/UX type name, includes the swap file-system type, and matches the `autorecovery` cluster number. When slice 2 is opened as a `usr` file system, the first partition that matches the A/UX type name, includes the identifying bit of a `usr` file system, and matches the `autorecovery` cluster number is associated. Any of these default file-system assignments may be overridden by explicitly setting a partition name for the partition. The partitioning should not be reset on an active file system.

Slice 31 is always assigned to the entire physical drive. Slice 31 cannot be reassigned to another partition.

If the partition-map information is missing from the beginning of the disk, the driver provides the following default mapping:

Partition 0	A partition that starts at block 204. Length is the entire disk minus the size of partition 1. This partition is usually used for a Root&Usr file system.
Partition 1	One quarter of the disk or up to 10 megabytes, whichever is less. This partition is normally used for `swap`.

Partition 31 Entire disk.

Only a subset of the following ioctls are allowed on these default partitions and they are the following: GD_PARTSIZE, GD_UNSETPNAME, UIOCEXTE, UIOCNEXTE, UIOCFORMAT, GD_SOFTERR, and GD_SPARE.

IOCTLS

The driver accepts the following ioctls (the symbol definitions are located in /usr/include/sys/ioctl.h or /usr/include/sys/ssioctl.h):

ioctl (*fd*, GD_ALTBLK, *bool*)

The alternate block mechanism sets aside a portion of each partition for an alternate block map area. When the boolean value used as the third argument to this function is TRUE, alternate block mapping occurs, and accesses are limited to the logical data area of the partition. When the third argument is FALSE, alternate block mapping is disabled, and reads and writes are allowed throughout the partition. This ioctl may only be performed by the superuser, or on a file descriptor that is open for writing.

ioctl (*fd*, GD_GBZBTMADE, 0)
ioctl (*fd*, GD_GBZBTMOUNT, 0)
ioctl (*fd*, GD_GBZBTUMOUNT, 0)

These ioctls return the value of the time field stored in the block 0 block of the partition. The times correspond to the time the file system was made, the last time the file system was mounted, and the last time the file system was unmounted. The ioctl returns type time_t (defined in /usr/include/types.h).

ioctl (*fd*, GD_GETABM, *addr*)

This ioctl returns a description of the alternate block map for the partition. The abm data structure (defined in /usr/include/apple/abm.h) is comprised of:

```
struct abm {
     int     abm_size;  /* size of map (bytes) */
     int     abm_ents;  /* number used (bytes) */
     daddr_t abm_start; /* start of map (blk num) */
};
```

If alternate block mapping is not applied to the partition, the system call errno is set to ENXIO.

`ioctl` (*fd*, `GD_GETMAP`, *abmi*)
This `ioctl` returns alternate block information for the partition. The return of data is controlled by a pointer to the `abmi` data structure passed as the third argument to the function. The `abmi` data structure is defined in `/usr/include/apple/abm.h`.

```
struct abmi {
      caddr_t  abmi_buf;     /* read buffer */
      int      abmi_nbytes;  /* read count */
};
```

`abmi_buf` is the location to place the alternate block map information. `abmi_nbytes` is the number of bytes to read. The size of the alternate block map may be determined by the `ioctl GD_GETABM`.

`ioctl` (*fd*, `GD_GETPNAME`, *dpident*)
`ioctl` (*fd*, `GD_SETPNAME`, *dpident*)

```
struct dpident {
       char  dpiname[32]; /* name of partition */
       char  dpitype[32]; /* type of partition */
};
```

These ioctls map named partitions (named by *dpident*) to A/UX devices (*fd*). The notion of partition mapping is discussed earlier. The name and type of a partition are character strings. A typical name would be `partition00`. The type name for A/UX file systems is defined as `Apple_UNIX_SVR2`. By specification, the names need not be null terminated if they are `DPISTRLEN` long (see `dpme`(4) for further information). The `ioctl` to get the name copies the current partition information corresponding to the file descriptor into the user's buffer. `errno` is set to `ENXIO` if there is no partition assigned or if there are no disk-partition-map entries for the disk. The `ioctl` to set the name searches the disk for the first partition that matches *dpident*, and assigns it to the major and minor device number corresponding to the file descriptor. The `dpident` structure is defined in `/usr/include/apple/dpme.h`.

The ioctl `GD_SETPNAME` may only be performed by the superuser, or on a file descriptor that is open for writing.

`ioctl` (*fd*, `GD_MKBAD`, *blocknum*)
The block number given as the third argument (*blocknum*) to the call is entered in the software-maintained alternate block

map for the partition. The block numbers are always relative to the start of the partition. An alternate block map must be created by application-formatting software before the block is added. This `ioctl` may only be performed by the superuser, or on a file descriptor that is open for writing.

`ioctl` (*fd*, `GD_SBZBTMADE`, *time*)
`ioctl` (*fd*, `GD_SBZBTMOUNT`, *time*)
`ioctl` (*fd*, `GD_SBZBTUMOUNT`, *time*)
These `ioctl`s set time fields stored in the block 0 block for the partition to the third argument (*time*) of the function. These ioctls may only be performed by the superuser, or on a file descriptor that is open for writing. The parameter *time* is type `time_t`.

`ioctl` (*fd*, `GD_PARTSIZE`, `0`)
The return value of the function is the size of the partition. This `ioctl` returns a `long` integer that represents the partition's logical size in blocks.

`ioctl` (*fd*, `GD_SHUTDOWN`, *arg*)
Special shutdown processing is performed. The argument is normally one of these values:

```
GD_SHUT_SHIP   1 /* Shutdown and ready for shipping*/
GD_SHUT_CLOSE  2 /* Internal driver information*/
GD_SHUT_REINIT 3 /* Reinitialize the driver*/
```

The first two values are implemented by device-specific software and may or may not have an effect on any given device. The last value causes drive data structures to be reinitialized and is used by utility software that updates disk-partition information.

`ioctl` (*fd*, `GD_SOFTERR`, *bool*)
If the third argument to this function is a Boolean value of TRUE, soft errors on the disk return as hard errors to applications. In general, this means that an error that could be corrected in hardware is not corrected and remains an error. For SCSI disks, this `ioctl` is only defined for devices that adhere to the common command set. Consult the manufacturer's documentation for non-SCSI disks. This `ioctl` is intended for Apple diagnostics and should not normally be used. The `ioctl` may only be used on a file descriptor corresponding to partition 31 of the disk. It may only be used by the superuser, or if the file descriptor is open

for writing.

`ioctl(`*fd*`, GD_SPARE,` *blocknum*`)`
This `ioctl` causes hardware-specific bad blocking of the block number given as the third argument (*blocknum*) to the function. The block number is always relative to the start of a partition. This `ioctl` call may only be used by the superuser, or if the file descriptor is open for writing.

`ioctl(`*fd*`, GD_UNSETPNAME, 0)`
The partition name, if any, assigned to the minor device number is removed. If the minor number is assigned a partition by default, the partition name assignment is recalculated on next access. Otherwise, an error (`ENXIO`) is returned on next access. This function may only be invoked by the superuser, or if the file descriptor is open for writing.

`ioctl(`*fd*`, UIOCEXTE, 0)`
This `ioctl` activates error printing on the system console. The file descriptor may correspond to any character device file associated with the desired controller. This `ioctl` may only be performed by the superuser, or on a file descriptor that is open for writing.

`ioctl(`*fd*`, UIOCFORMAT,` *diskformat*`)`
This `ioctl` formats the disk. The placeholder *fd* should be an open file descriptor of the character device (that is, `/dev/rdsk/c?d?s?`). The third parameter to the `ioctl` function call contains the address of a `diskformat` structure (defined in `/usr/include/sys/diskformat.h`). The `d_secsize` field of this structure may be used to specify 512-byte or 532-byte sector size for formatting. The 532-byte sector size will not be supported in future versions of the driver and should not be used. The other fields of the diskformat structure are ignored. The `ioctl` may only be applied to partition 31 of a disk. The `ioctl` may only be used by the superuser, or if the file descriptor is open for writing.

`ioctl(`*fd*`, UIOCNEXTE, 0)`
This `ioctl` deactivates error printing. The file descriptor may correspond to any character device file associated with the desired controller. When deactivated, device errors continue to be logged by `errdemon`(1M) but do not disturb the display on the console. This `ioctl` may only be performed

by the superuser, or on a file descriptor that is open for writing.

ERRORS

The following error values may be returned:

[ENXIO]	The device or partition could not be found.
[EACCES]	The requested `ioctl` is only permitted by the superuser, or on a file open for writing.
[EINVAL]	An `ioctl` to perform alternate block or partition manipulation was performed on a disk lacking appropriate disk partition map entries.
[EBUSY]	An attempt was made to set the partition name of a minor device that is already in use.
[EEXIST]	An attempt was made to set the partition name of a partition already assigned to another minor device.
[EIO]	An I/O error has occurred.
[ENOSPC]	There is no space in the software alternate block map.

FILES

```
/dev/dsk/c?d[0-7]s*
/dev/rdsk/c?d[0-7]s*
/usr/include/sys/gdisk.h
/usr/include/sys/ioctl.h
/usr/include/sys/ssioctl.h
/usr/include/apple/abm.h
```

SEE ALSO

exterr(1), badblk(1M), diskformat(1M), dp(1M), mkfs(1M), mknod(1M), pname(1M), ioctl(2), getptabent(3), altblk(4), bzb(4), dpme(4), ptab(4), autorecovery(8), boot(8), StartupShell(8).

kmem(7)

See mem(7)

NAME

mem, kmem — an interface for access to core memory

DESCRIPTION

mem is a special file that is an image of the core memory of the computer. You can use it, for example, to examine and even patch the system.

Byte addresses in mem are interpreted as memory addresses. References to non-existent locations return errors.

Examining and patching device registers is likely to lead to unexpected results when there are read-only or write-only bits.

The file kmem is the same as mem, except that it accesses kernel virtual memory rather than physical memory.

FILES

/dev/mem
/dev/kmem

NAME

mouse — mouse input device driver

DESCRIPTION

The mouse driver provides simple access to the mouse device. This driver allows one to open the mouse device, read the mouse position and sense the mouse button state. All I/O transactions complete immediately, whether succeeding or not.

Opens of /dev/mouse will always succeed as long as no other user is using the mouse (see console(7)) and the mouse is connected to the system. Reads from the mouse device always return 4 bytes. The first two are a 16 bit signed absolute mouse horizontal position. The next two bytes give the vertical position in a similar manner. Positioning is relative to (0, 0) when the A/UX system was started.

Writes to the mouse device always fail. The mouse supports one ioctl, MOUSE_BUTTON, which takes as a parameter the address of a character into which it returns the current state of the mouse's button. The symbol for this ioctl is located in /usr/include/sys/mouse.h.

FILES

/dev/mouse
/usr/include/sys/mouse.h

SEE ALSO

console(7).

NAME

mtio — interface conventions for magnetic tape devices

DESCRIPTION

The mtio library allows applications to access tape devices using a standard interface.

While the general read, write, and seek operations are part of the standard I/O library, additional library resources are required to add support for tape devices. The mtio library helps provide a fairly device-independent way to perform additional operations, such as rewind, that are common to tape drives, but not common to other types of devices.

There are several system-dependent parameters that this generic device interface uses. The most important is a standard I/O block size, BLKDEV_IOSIZE, which is defined in <sys/param.h>. For A/UX, this value is 1024. The commands that typically are used in conjunction with a tape drive, such as tar(1), ultimately generate requests to read and write 1024-byte blocks when the tape devices are specified for either input or output. However, a peculiarity of the Apple SC 40 Tape Backup is that it can only read and write in blocks of 8192 bytes, requiring a specially formulated user command line (see tc(7) and tcb(1)).

The mtio library moderates device I/O when output is sent to or obtained from the device files in /dev/rmt.

IOCTLS

When manipulating the tape cartridge device more directly through your own programs, or through programs such as mt(1), tape-specific operations are supported through mtio ioctls. With a few exceptions, these generic operations are shared by all tape drives. One exception is the "format" ioctl (MTFORMAT), which is available for the Apple SC 40 Tape Backup only; in general, this is not a function of reel tape drives, such as 9-track tape drives. Other exceptions that apply to the tape cartridge unit are described in tc(7). The following is a list of mtio ioctls, and their associated subcommands. For complete details about the data structures, see <sys/mtio.h>.

ioctl(*fd*, MTIOCGET, **mtget*)

This tape ioctl causes the tape unit to return the status of the tape drive, returning values in *mtget*.

`ioctl` (*fd*, `MTIOCTOP`, `*`*mtop*)

This tape ioctl causes the tape unit to peform the subcommand selected by the current values stored in the structure *mtop*.

The subcommands for `MTIOCTOP` include:

`MTWEOF`	write an end-of-file record
`MTFSF`	forward space file
`MTBSF`	backward space file
`MTFSR`	forward space record
`MTBSR`	backward space record
`MTREW`	rewind
`MTOFFL`	rewind and put the drive offline
`MTNOP`	no operation, sets status only

FILES

```
/usr/lib/sys/ioctl.h
/usr/lib/sys/mtio.h
/dev/rmt/*
```

SEE ALSO

`mt`(1), `tar`(1), `tcb`(1), `tp`(1), `tc`(7).

NAME

null — the null device file

DESCRIPTION

Data written on the null device file, /dev/null, are discarded.

Reads from /dev/null always return 0 bytes.

FILES

/dev/null

NAME

nvram — nonvolatile memory/time of day clock interface

DESCRIPTION

The device /dev/nvram provides access to the real time clock chip on the Macintosh II system board. This chip contains 256 bytes of nonvolatile memory (memory that retains its contents when the system's power is turned off). It also contains a time of day clock.

The special device /dev/nvram can be read by anyone. It returns up to 256 bytes. The bytes are the contents of the nonvolatile RAM. In order to write to /dev/nvram, you must be the superuser (root).

If the device is already open, an attempt to open it will return the error EBUSY. This is to ensure that a process can perform read-modify-write operations on the device. If an open fails with EBUSY, the process should wait a while and then try again. After an open succeeds, the device should be kept open only as long as is necessary.

WARNINGS

Care should be taken in writing this device. Since the contents of the nonvolatile RAM are defined by Apple and used by many parts of both A/UX and the Macintosh operating system, indiscriminate writing could cause your system to malfunction. Usually you should use the utilities provided to manipulate the nvram contents.

FILES

/dev/nvram
/usr/include/sys/nvram.h

SEE ALSO

date(1), stime(2), time(2).

NAME

pty — pseudo terminal driver

DESCRIPTION

The pty driver provides support for a device-pair termed a *pseudo terminal*. A pseudo terminal is a pair of character devices, a *master* device and a *slave* device. The slave device provides processes an interface identical to that described in termio(7). However, whereas all other devices which provide the interface described in termio(7) have a hardware device of some sort behind them, the slave device has, instead, another process manipulating it through the master half of the pseudo terminal. That is, anything written on the master device is given to the slave device as input and anything written on the slave device is presented as input on the master device.

The following ioctl calls apply only to pseudo terminals:

TIOCPKT — Enable/disable "packet" mode. Packet mode is enabled by specifying (by reference) a nonzero parameter and disabled by specifying (by reference) a zero parameter. When applied to the master side of a pseudo terminal, each subsequent read from the terminal will return data written on the slave part of the pseudo terminal preceded by a zero byte (symbolically defined as TIOCPKT_DATA), or a single byte reflecting control status information. In the latter case, the byte is an inclusive-or of zero or more of the bits:

TIOCPKT_FLUSHREAD
: whenever the read queue for the terminal is flushed.

TIOCPKT_FLUSHWRITE
: whenever the write queue for the terminal is flushed.

TIOCPKT_STOP
: whenever output to the terminal is stopped as with (CONTROL-S).

TIOCPKT_START
: whenever output to the terminal is res-

tarted.

TIOCPKT_DOSTOP
whenever t_stopc is CONTROL-S and t_startc is CONTROL-Q.

TIOCPKT_NOSTOP
whenever the start and stop characters are not CONTROL-S/CONTROL-Q.

This mode is used by rlogin(1N) and rlogind(1M) to implement a remote-echoed, locally CONTROL-S/CONTROL-Q flow-controlled remote login with proper back-flushing of output; it can be used by other similar programs.

FILES

/dev/pty[p-r][0-9a-f]
/dev/tty[p-r][0-9a-f]

BUGS

It is not possible to send an EOT.

NAME

serial — the on-board serial ports

DESCRIPTION

/dev/tty0 is the serial port connected to the DIN connector on the rear of the chassis with the modem icon above it; it is linked to (is the same as) the name /dev/modem. /dev/tty1 is the serial port connected to the DIN connector on the rear of the chassis with the printer icon above it; it is linked to (is the same as) the name /dev/printer.

These ports support all the standard A/UX ioctls from termio(7). They also support the following hardware specific extensions. The modes set by these ioctls (or the corresponding stty(1) options) persist after a device is closed and reopened. The mnemonic definitions are in /usr/include/sys/ioctl.h.

UIOCNOMODEM — No modem control, the input line HSKi is ignored. The output line HSKo is asserted whenever the line is opened. The following stty(1) command can be used to put a port (/dev/tty0 in this example) into such a mode:

```
stty -modem < /dev/tty0
```

UIOCMODEM — modem control, the output line HSKo is asserted whenever a process is attempting to open the device or while the device is open, as an output this performs the RS232 function Data Terminal Ready (DTR). When a port is closed this line is negated (if the HUPCL flag from termio(7) is set) which causes a modem to hang up a call). Upon opening, if HSKi is not asserted, then an open will not complete (a process will be suspended until the open does complete) until it is asserted. Processes that use the O_NDELAY (see open(2)) flag when opening such a port are not suspended, but instead complete their opening immediately. If HSKi is negated while a port is open,

the signal SIGHUP will be generated to processes with the port as their controlling terminal (usually resulting in the death of the processes and the subsequent closing of the port). In this mode, the HSKi input functions as the RS232 function Data Carrier Detect (DCD). The following stty(1) command can be used to put a port into such a mode:

```
stty modem < /dev/tty0
```

To turn this option off, stty(1) needs to first open the port; and because it can't open the port until the HSKi (DCD) line is asserted, the following form of the stty(1) should be used to turn off this option:

```
stty -n /dev/tty0 -modem
```

UIOCDTRFLOW

DTR flow control. This mode is used to communicate with printers such as the Apple ImageWriter II. The HSKo output performs in the same manner as above. The HSKi input is used to enable or disable output by a a device (such as a printer) that wishes to flow control it. When HSKi is asserted characters may be output, when it is negated output will stop. Note that when an A/UX device is closed, the process that is closing it will be suspended until all waiting characters have been transmitted, such a process will wait until flow control is asserted. Either of the following stty(1) commands can be used to turn on DTR flow control:

```
stty dtrflow < /dev/tty0
stty hxctl < /dev/tty0
```

`UIOCTTSTAT` This call returns the current state of the three above options, refer to `termio(7)` for more documentation.

Note: Since this interface does not support the RS232 "Clear To Send" and "Request To Send" (CTS/RTS), the ioctls `UIOCFLOW` and `UIOCNOFLOW` are not supported. In `UIOCDTRFLOW` mode, the HSKi input acts very similarly to CTS and could be used with some devices that require this signal.

The port pins have the following functions, shown with the connections required to use them for RS232:

Mini-DIN Pin #	Function	RS232	Pin #
1	HSKo	DTR	20
2	HSKi	DCD	8
3	TXD-	TXD	3
4	GND	GND	7
5	RXD-	RXD	2
6	TXD+	GND	7
7	GPi	NC	
8	RXD+	GND	7

FILES

```
/dev/tty0
/dev/tty1
/dev/modem
/dev/printer
/usr/include/sys/ioctl.h
```

SEE ALSO

`stty(1)`, `ioctl(2)`, `open(2)`, `termio(7)`.

NAME

streams — an interface for character I/O

DESCRIPTION

Streams is a mechanism that is used in the UNIX kernel for some device drivers. These drivers are usually for communications or tty type applications. To most programs, this interface is, with a few exceptions, the same as that of traditional UNIX character devices. When used with a streams line-discipline, this interface is the same as that of normal UNIX terminals.

A/UX supports the version of streams implemented under UNIX V.2.1, which is a functional subset of that provided by later UNIX implementations. It is upwardly compatible with such systems.

The main difference between streams and other character device drivers is that the streams interface is message based. Commands and data exchanged between devices, processes and line disciplines (streams modules) are sent in messages. A number of special ioctl functions have been defined to send and receive these messages.

A stream is built by opening a stream style device (the ability of a device to "stream" is defined by the writer of the device's device-driver, either a device streams or it doesn't). When the device is open it consists of the device and the "stream head," the interface to the process that opened the device.

The device and the stream head can communicate by means of messages across the full-duplex stream. Processes can communicate with the stream head by means of system calls such as read(2), write(2) and ioctl(2).

Using the I_PUSH, a stream module can be pushed (in a LIFO or stacked manner) onto the stream. More than one stream module can be pushed onto such a stream at a time. The module closest to the stream head may be removed using the I_POP ioctl call.

Closing a stream causes the modules to be popped from the stream, the device to be closed, and the stream dismantled.

Modules exist in the kernel and are referenced by name. There are two standard streams modules:

line — A tty style line discipline. When pushed, it implements all the functionality described by termio(7). Most terminal-style communications

lines use this module.

`shlr` The shell layering module. `shlr` responds to shell layering ioctls to implement shell layering on stream based ttys. Normally `shl`(1) is the only utility that uses this module.

The streams system implements a number of ioctls, all of the ioctls described in `termio`(7) are provided for compatability. Some devices and/or modules may not respond to these calls. In particular many of the line discipline related ioctls will either fail or be ignored unless the module `line` has been pushed onto the stream. In addition the following streams related ioctls are supported, they are defined in the include file <`sys/stropts.h`>.

`I_STR` `ioctl(fd, I_STR, &strioctl)`
`struct strioctl strioctl;`

This ioctl builds an ioctl packet and sends it down the stream. It may be interpreted by any module on the stream or by the device at the end. The packet is returned with data and an indication of success or failure. The data structure `strioctl` is used to describe the packet to be sent. It has 4 fields:

`ic_cmd`	The command to be sent
`ic_timout`	How long to wait for the ioctl to succeed before failing (in seconds), values 0 and -1 have special meanings, 0 means wait for the system default time, -1 means wait forever.
`ic_dp`	Points to the address of the data to be sent down the stream, or the address at which data returned from the stream is to be stored.
`ic_len`	Is the length of the data to be sent, in bytes, or the size of the buffer into which returned data is to be stored.

Errors:
EFAULT if ic_dp references an invalid address. Any other device/module specific error message.

I_NREAD

```
n=ioctl(fd, I_NREAD, &first);
```

I_NREAD returns the number of messages in the queue at the streams head as its result. It also returns the number of bytes in the first message in the queue to the address referenced by its argument.

Errors:
EFAULT if the argument references an invalid address.

I_PUSH

```
ioctl(fd, I_PUSH, module)
char *module;
```

This ioctl pushes the streams module named by the null-terminated string module onto an open stream.

Errors:
EFAULT if the module name references an invalid address. EINVAL if the module name does not describe an existing streams module in the A/UX kernel.
Any other error the streams module might return if it decides not to allow the push.

I_POP

```
ioctl(fd, I_POP, 0)
```

The ioctl removes the streams module closest to the process on the stream.

Errors:
EINVAL if no such module exists

I_LOOK

```
ioctl(fd, I_LOOK, buff)
char buff[FMNAMESZ+1];
```

This returns the name of the streams module closest to the process on a stream.

Errors:
EFAULT if the buffer for the name is located at an invalid address.
EINVAL if there are no modules pushed onto the stream.

I_FLUSH `ioctl(fd, I_FLUSH, flushtype)`

This generates a message that is sent down the queue to flush messages waiting at modules down the stream. The parameter can be one of three allowed values:

FLUSHR	flush messages coming down the stream towards the process
FLUSHW	flush messages moving up the stream away from the process
FLUSHRW	flush all messages in the stream

Errors:
EINVAL the parameter is not one of the above values
EAGAIN insufficient resources are available to send the message up the queue and it should be retried at a later time.

I_SRDOPT `ioctl(fd, I_SRDOPT, srdtype)`

This ioctl changes the manner in which the stream head treats incoming messages as they are passed to a process as part of a read(2) system call. The parameter can take one of three possible values:

RNORM	in stream mode - messages are read from the stream and message boundaries are ignored (except for 0 length messages which are always returned as separate messages and are

normally treated as end of file markers)

RMSGN — a read terminates at either the end of the message or when the read buffer is full. Any message data remaining is available from future reads.

RMSGD — a read terminates when either the end of the message is found or the read buffer is full. Any unread data is discarded.

When a stream is first opened, it has the default operating mode of RNORM. It is also possible for upstream modules to change this.

Errors:
EINVAL the parameter is not one of the above values

I_GRDOPT

```
ioctl(fd, I_GRDOPT, &opt)
int opt;
```

This call returns the current read option (as specified above under I_SRDOPT).

Errors:
EFAULT if the argument is a valid address

I_FIND

```
find = ioctl(fd, I_FIND, buff)
char buff[FMNAMESZ+1];
```

This call returns 1 if a module of the name given in the null-terminated string passed in the argument is present in the stream, or 0 if the module does not exist.

Errors:
EINVAL if the name is not the name of a module in the kernel.
EFAULT if the address of the name passed as

the argument is not valid.

I_MNAME

```
ioctl(fd, I_MNAME, &par)
union {
int depth;
char buff[FMNAMESZ+1];
} par;
```

This ioctl returns the name of the module or driver at the depth on the stream specified by the parameter. The stream head is at depth 0. The last module found on the stream will be the driver.

Note: this ioctl is not necessarily provided on all systems that provide a stream interface. It should not be used if program portability is a factor.

Errors:
EFAULT if the address of the parameter is invalid
EINVAL if the depth is less than 0 or references a module past the driver at the end of the stream

Terminal Lines

When using a stream based terminal, it is usually necessary to push a line discipline module onto the stream before use. In almost all cases this is done by /etc/getty or /etc/init when you log onto your system. When you are opening an unused terminal line it may then be required. Pushing more than one line discipline onto a stream should be avoided as the results are undefined and will not be useful. Two methods are provided to make pushing line disciplines easier. They both can be used on non stream based character drivers without any undue effect and they will avoid the multiple pushing of line disciplines if one is already pushed.

line_push(3) is a library routine that is passed the file descriptor of and open device (from open(2)). It will push a line discipline onto the device if it is a streaming device and there is not one pushed already.

/etc/line_sane(1M) is a utility that can be run from shell scripts (such as those started from /etc/inittab(4)). It takes one parameter, an integer representing an open file descriptor on which the line discipline is to be pushed. It behaves similarly to line_push(3) above.

Further Functionality

The following extensions are provided to the streams system. They are not necessarily provided with other streams implementations and should not be used if portability is important.

select(2)	allows a process to wait for input from more than one open device, socket or stream
FIONBIO	allows a process to make non blocking reads to a stream (see termio(7))
FIONASYNC	sends a SIGIO signal to a process when input is available from the queue (see termio(7))
FIONREAD	returns the number of characters available to be read from the stream head (this is different from I_NREAD above) (see termio(7))

SEE ALSO

line_sane(1M), close(2), ioctl(2), open(2), read(2), select(2), write(2), line_push(3).
Building A/UX Device Drivers.

NAME

sxt — pseudo-device driver

DESCRIPTION

sxt is a pseudo-device driver that interposes a discipline between the standard tty line disciplines and a real device driver. The standard disciplines manipulate *virtual tty* structures (channels) declared by the sxt driver. sxt acts as a discipline manipulating a *real tty* structure declared by a real device driver. The sxt driver is used only by the shl(1) command.

Virtual ttys are named by inodes in the subdirectory /dev/sxt and are allocated in groups of up to eight. To allocate a group, a program should exclusively open a file with a name of the form /dev/sxt/??0 (channel 0) and then execute a SXTIOCLINK ioctl call to initiate the multiplexing.

Only one channel, the *controlling* channel, can receive input from the keyboard at a time; others attempting to read will be blocked.

sxt supports two groups of ioctl(2) commands. The first group contains the standard ioctl commands described in termio(7), with the following additions:

TIOCEXCL — Set *exclusive use* mode: permit no further opens until the file is closed.

TIOCNXCL — Reset *exclusive use* mode: permit further opens.

The second group, which follows, are directives to sxt itself. Some of these may only be executed on channel 0.

SXTIOCLINK — Allocate a channel group and multiplex the virtual ttys onto the real tty. The argument is the number of channels to allocate. This command may only be executed on channel 0. Possible errors include

- EINVAL The argument is out of range.
- ENOTTY A real tty did not issue the command.
- ENXIO sxt did not configure *linesw*.
- EBUSY Already issued an SXTIOCLINK command for

this real *tty*.

ENOMEM There is no system memory available for allocating the virtual tty structures.

EBADF Did not open channel 0 before this call.

SXTIOCSWTCH Set the controlling channel. Possible errors include

EINVAL Gave an invalid channel number.

EPERM Did not execute the command from channel 0.

SXTIOCWF Cause a channel to wait until it is the controlling channel. This command returns the error EINVAL if it is given an invalid channel number.

SXTIOCUBLK Turn on the loblk control flag in the virtual tty of the indicated channel.

SXTIOCUBLK Turn off the loblk control flag in the virtual tty of the indicated channel. Returns the error EINVAL if it is given an invalid number or channel 0.

SXTIOCSTAT Get the status (blocked on input or output) of each channel and store the argument references in the sxtblock structure. Returns the error EFAULT if it cannot write the structure.

SXTIOCTRACE Enable tracing. This command has no effect if tracing is not configured.

SXTIOCNOTRACE Disable tracing. This command has no effect if tracing is not configured.

If the device driver is a streams device driver (see streams(7)) then it will respond to all the sxt ioctls listed earlier, provided they are made using the streams I_STR ioctl. To turn on shell layering on a streams device, first pop the normal line discipline off the stream, push the line discipline shlr onto the stream, and then push the line discipline line back onto the stream. This

open file descriptor will now become the controlling virtual tty. To open a slave virtual tty, open `/dev/shl` and push a line discipline onto it (using `line_push`(3), for example). `/dev/shl` is a clone device so a new virtual device will automatically be opened exclusively for you (`TIOCEXCL` is not required or supported).

FILES

`/dev/sxt/??[0-7]`	virtual tty devices
`/usr/include/sys/sxt.h`	driver-specific definitions
`/dev/shl`	

SEE ALSO

`shl`(1), `stty`(1), `ioctl`(2), `open`(2), `line_push`(3), `streams`(7), `termio`(7).

NAME

tc — Apple Tape Backup 40SC device driver

DESCRIPTION

tc is a device driver that supports the Apple Tape Backup 40SC using the generic tape driver interface described in mtio(7), with certain exceptions. To be more compatible with 9-track tape units, this driver approximates 9-track drive tapemarks through the use of special tapemark records. Other exceptions are mentioned where appropriate.

The Apple Tape Backup 40SC is connected to the system through the SCSI device chain. To select the correct device, you must reference the correct device file. The SCSI ID number of the tape backup device should match the number after tc in the device file. The device files for Apple tape backup devices in the form

```
/dev/rmt/tcx
/dev/rmt/tcxn
```

where *x* is the SCSI ID number of the drive in the range 0 through 7. For example, /dev/rmt/tc0 and /dev/rmt/tc0n both select the device with a SCSI ID set to 0. The device reference with suffix n is called a reference to a no-rewinding tape device because it reads or writes from the tape drive without rewinding the tape afterwards. If you use the no-rewinding device, the tape is positioned so that the next archive in a series of archives can be accessed. (These device files are present only if the kernel has been configured properly for the Apple Tape Backup 40SC.)

To configure the kernel, run the command

```
newconfig tc
```

to create a kernel with tape-cartridge support. Note that newunix must first be run, to enable creation of device files.

The tape cartridge must be formatted in order to store data. Nominal capacity is 40 megabytes (MB). Usable capacity is closer to 38.5 MB (reduced because of formatting overhead).

One of the peculiarities of this tape device is that it only reads and writes fixed 8 kilobyte (KB) blocks, streaming when possible. Since the device always reads and writes 8 KB blocks, the driver restricts raw I/O size to a multiple of 8 KB. Reading and writing many 8 KB blocks at once minimizes user-process overhead and maximizes streaming. It should be noted that positioning subcommands always act on physical 8 KB blocks.

The tcb(1) program helps overcome the fixed-block-size problem when it is used to read from the tape or when it is used just before a write to the tape.

Each read or write call reads or writes the next record on the tape. When writing, the record size returned is the same length as the block size given. When reading, the record size returned is the actual number of bytes read, which can be no greater than the buffer size; if the record is too long, an error is indicated.

For the tc device driver, the ioctls and data structures associated with the mtio(7) generic tape interface are supported, including MTIOCGET and MTIOCTOP. The MTIOCGET ioctl returns the driver's notion of current file and block number information in mt_fileno and mt_blkno. The mt_dsreg field contains the logical block number where the last I/O operation occurred. Though this generally matches mt_blkno, they are both returned for testing purposes.

The mt_erreg field always contains the current driver version. The number is a short integer, which is expressed as *x.mn* (for example, 3.24). The version is encoded as

$$(100 * x) + (10 * m) + n$$

For this example, the value would be decimal 324.

The mt_resid field contains the number of usable blocks on the currently loaded cartridge. If this field is 0, the cartridge is not formatted.

FILES

/dev/rmt/tc[0-7]	Rewind on close
/dev/rmt/tc[0-7]n	No rewind on close

SEE ALSO

mt(1), tar(1), tcb(1), tp(1), mtio(7).

DIAGNOSTICS

If an unformatted cartridge is loaded and the device is opened for read only, EINVAL is returned.

BUGS

The tapemark format does not conform to any known standard because such standards are nonexistent. When end-of-file is encountered at a simulated tapemark, the tapemark record is transferred to the user's buffer, even though the return from read(2) (correctly) does not include the tapemark bytes. A 0-byte count is

returned when a tapemark is read, but another read fetches the first record of the next tape file.

If the tape device remains in an error state, `tc` may have to be closed to clear the error condition.

The `mtio`(7) subcommands `MTNOP`, `MTCACHE`, and `MTNOCACHE` are not supported.

After issuing the `MTFORMAT` ioctl, you must close and reopen the device before it will allow any other subcommands.

For raw tape I/O accesses, seeks are ignored.

NAME

termio — general terminal interface

DESCRIPTION

This section describes both a particular file and the terminal interface.

The file /dev/tty is, in each process, the control terminal associated with the process group of that process. Programs or shell sequences use it to ensure that their messages appear on the terminal, no matter how output is redirected. Also, programs that demand an output filename will accept /dev/tty, so the terminal being used is unimportant.

The asynchronous communications ports use the same general interface, no matter what their hardware. This section discusses the common features of this interface.

When a terminal file is opened, it normally makes the process wait until it establishes a connection. Users' programs seldom open these files; getty(1M) opens them and they become a user's standard input, output, and error files. The first terminal file the process-group leader opens, which is not already associated with a process group, becomes the *control terminal* for that process group. The control terminal plays a special role in handling quit and interrupt signals, as discussed later. The control terminal is inherited by a child process during a fork(2). A process breaks this association by changing its process group (using setpgrp(2)).

Terminals associated with one of these files operate in full-duplex mode. You may type at any time, even while the terminal is printing. Characters you type are lost only when the system's character input buffers are full, which is rare, or when you have accumulated the maximum number of input characters that have not been read by some program. Currently, this limit is 256 characters. When you reach the input limit, all the saved characters are thrown away without notice.

Normally, terminal input is processed in units of lines. A line is delimited by a newline (ASCII LF) character, an end-of-file (ASCII EOT) character, or an end-of-line character. This means that a program cannot read input until you have typed an entire line. Also, no matter how many characters a read(2) system call requests, a maximum of one line is returned. It is not, however, necessary to read a whole line at once; a read can request any

number of characters, even one, without losing information.

Erase and kill processing is normally done during input. By default, the character DELETE erases the last character typed, but it does not erase beyond the beginning of the line. By default, the character CONTROL-U deletes the entire input line, and optionally outputs a newline character. Both these characters operate on a key-stroke basis, independent of any backspacing or tabbing that may have been done. You can escape both the erase and kill characters by preceding them with the escape character (\). The user can also change the erase and kill characters with `stty`(1).

The following characters have special input functions.

INTR (Rubout or CONTROL-C) Interrupt signal to all processes associated with the control terminal. Normally, it terminates each process, but you can arrange to have it ignore the signal or to receive a trap to an agreed-upon location; see `signal`(3).

SWTCH (CONTROL-Z or ASCII SUB) Used by the shell layering facility, `shl`, to change the current layer to the control layer.

QUIT (CONTROL-\ or ASCII FS) Generates a quit signal. It is identical to the interrupt signal except that, unless a receiving process has made other arrangements, it will also create a core image file (called `core`) in the current working directory.

ERASE (DELETE) Erases the preceding character. It will not erase beyond the start of a line, as delimited by a NL, EOF, or EOL character.

KILL (CONTROL-U) Deletes the entire line, as delimited by a NL, EOF, or EOL character.

EOF (CONTROL-D or ASCII EOT) Generates an end-of-file from a terminal. This passes the characters waiting to be read to the program, without waiting for a newline, and discards the EOF. If no characters are waiting, (if EOF occurred at the beginning of a line), 0 characters are passed back; this is the standard end-of-file indication.

NL (ASCII LF) The normal line delimiter. It cannot be changed or escaped.

EOL (ASCII NUL) An additional line delimiter, like NL. It is not normally used.

STOP (CONTROL-S or ASCII DC3) Temporarily suspends output. It is useful for preventing output from disappearing from CRT terminals before you have read it. While output is suspended, STOP characters are ignored and not read.

START (CONTROL-Q or ASCII DC1) Resumes output suspended by a STOP character. While output is not suspended, START characters are ignored and not read. The start/stop characters cannot be changed or escaped.

Special character functions can be disabled by changing the value in `c_cc` of the `termio` structure to `'\0377'`.

The user can change the character values for INTR, QUIT, SWTCH, ERASE, KILL, EOF, and EOL using `stty`(1). You can escape the ERASE, KILL, and EOF characters by preceding them with a \ character, in which case no special function is generated.

When a data-set drops the carrier signal, a hangup signal is sent to all processes that have this terminal as the control terminal. Unless you have made other arrangements, this signal terminates the processes. If the hangup signal is ignored, subsequent reads return with an end-of-file indication. Thus, programs that read a terminal and test for end-of-file can terminate appropriately when hung up on.

When one or more characters are written, they are transmitted to the terminal as soon as previously-written characters have finished typing. Input characters are echoed by putting them in the output queue as they arrive. If a process produces characters more rapidly than they can be typed, it will be suspended when its output queue exceeds some limit. When the queue has drained down to some threshold, the program is resumed.

Several `ioctl`(2) system calls apply to terminal files. The primary calls use the following structure, defined in `termio.h`.

```
#define   NCC   8
struct termio {
      unsigned  short   c_iflag;   /* input modes */
      unsigned  short   c_oflag;   /* output modes */
      unsigned  short   c_cflag;   /* control modes */
```

```
        unsigned  short   c_lflag;   /* local modes */
        char      c_line;             /* line discipline */
        unsigned  char    c_cc[NCC]; /* control chars */
};
```

The special control characters are defined by the array `c_cc`. The relative positions and initial values for each function are as follows:

0	VINTR	^C
1	VQUIT	FS
2	VERASE	DEL
3	VKILL	^U
4	VEOF	EOT
5	VEOL	NUL
6	reserved	
7	SWTCH	NUL

The `c_iflag` field describes the basic terminal input control.

IGNBRK	0000001	Ignore break condition.
BRKINT	0000002	Signal interrupt on break.
IGNPAR	0000004	Ignore characters with parity errors.
PARMRK	0000010	Mark parity errors.
INPCK	0000020	Enable input parity check.
ISTRIP	0000040	Strip character.
INLCR	0000100	Map NL to CR on input.
IGNCR	0000200	Ignore CR.
ICRNL	0000400	Map CR to NL on input.
IUCLC	0001000	Map uppercase to lowercase on input.
IXON	0002000	Enable start/stop output control.
IXANY	0004000	Enable any character to restart output.
IXOFF	0010000	Enable start/stop input control.

If IGNBRK is set, the break condition (a character framing error with all data zeros) is ignored, that is, not put in the input queue and therefore not read by any process. Otherwise if BRKINT is set, the break condition will generate an interrupt signal and flush both the input and output queues. If neither IGNBRK nor BRKINT is set, a break condition is read as a NUL (0), or if PARMRK is set, as \377, \0, \0. If IGNPAR is set, characters with other framing and parity errors are ignored.

If PARMRK is set, a character with a framing or parity error that is not ignored is read as the three-character sequence: 0377, 0, X, where X is the data of the character received in error. To avoid ambiguity in this case, if ISTRIP is not set, a valid character of

0377 is read as 0377, 0377. If PARMRK is not set, a framing or parity error that is not ignored is read as the character NUL (0).

If INPCK is set, input parity checking is enabled. If INPCK is not set, input parity checking is disabled. This allows output parity generation without input parity errors.

If ISTRIP is set, valid input characters are first stripped to 7-bits, otherwise all 8-bits are processed.

If INLCR is set, a received NL character is translated into a RETURN character. If IGNCR is set, a received RETURN character is ignored (not read). Otherwise if ICRNL is set, a received RETURN character is translated into a NL character.

If IUCLC is set, a received uppercase alphabetic character is translated into the corresponding lowercase character.

If IXON is set, start/stop output control is enabled. A received STOP character will suspend output and a received START character will restart output. All start/stop characters are ignored and not read. If IXANY is set, any input character will restart output which has been suspended.

If IXOFF is set, the system will transmit START/STOP characters when the input queue is nearly empty/full.

The initial input control value is all-bits-clear.

The c_oflag field specifies the system treatment of output.

OPOST	0000001	Postprocess output.
OLCUC	0000002	Map lowercase to uppercase on output.
ONLCR	0000004	Map NL to CR-NL on output.
OCRNL	0000010	Map CR to NL on output.
ONOCR	0000020	No CR output at column 0.
ONLRET	0000040	NL performs CR function.
OFILL	0000100	Use fill characters for delay.
OFDEL	0000200	Fill is DEL, otherwise NUL.
NLDLY	0000400	Select newline delays:
NL0	0	
NL1	0000400	
CRDLY	0003000	Select carriage-return delays:
CR0	0	
CR1	0001000	
CR2	0002000	
CR3	0003000	
TABDLY	0014000	Select horizontal-tab delays:

TAB0	0	
TAB1	0004000	
TAB2	0010000	
TAB3	0014000	Expand tabs to spaces.
BSDLY	0020000	Select backspace delays:
BS0	0	
BS1	0020000	
VTDLY	0040000	Select vertical-tab delays:
VT0	0	
VT1	0040000	
FFDLY	0100000	Select form feed delays:
FF0	0	
FF1	0100000	

If OPOST is set, output characters are postprocessed as indicated by the remaining flags, otherwise characters are transmitted without change.

If OLCUC is set, a lowercase alphabetic character is transmitted as the corresponding uppercase character. This function is often used in conjunction with IUCLC.

If ONLCR is set, the NL character is transmitted as the CR-NL character pair. If OCRNL is set, the CR character is transmitted as the NL character. If ONOCR is set, no CR character is transmitted when at column 0 (first position). If ONLRET is set, the NL character is assumed to do the return function; the column pointer will be set to 0 and the delays specified for CR will be used. Otherwise the NL character is assumed to do just the line-feed function; the column pointer will remain unchanged. The column pointer is also set to 0 if the CR character is actually transmitted.

The delay bits specify how long transmission stops to allow for mechanical or other movement when certain characters are sent to the terminal. In all cases a value of 0 indicates no delay. If OFILL is set, fill characters will be transmitted for delay instead of using a timed delay. This is useful for high baud terminals which need only a minimal delay. If OFDEL is set, the fill character is DEL, otherwise NUL.

If a form feed or vertical tab delay is specified, it lasts for about 2 seconds.

Newline delay lasts about 0.10 seconds. If ONLRET is set, the return delays are used instead of the newline delays. If OFILL is set, two fill characters will be transmitted.

Return delay type 1 is dependent on the current column position, type 2 is about 0.10 seconds, and type 3 is about 0.15 seconds. If OFILL is set, delay type 1 transmits two fill characters, and type 2, four fill characters.

Horizontal tab delay type 1 is dependent on the current column position. Type 2 is about 0.10 seconds. Type 3 specifies that tabs are to be expanded into spaces. If OFILL is set, two fill characters will be transmitted for any delay.

Backspace delay lasts about 0.05 seconds. If OFILL is set, one fill character will be transmitted.

The actual delays depend on line speed and system load.

The initial output control value is all bits clear.

The c_cflag field describes the hardware control of the terminal:

CBAUD	0000017	Baud rate:
B0	0	Hang up
B50	0000001	50 baud
B75	0000002	75 baud
B110	0000003	110 baud
B134	0000004	134.5 baud
B150	0000005	150 baud
B200	0000006	200 baud
B300	0000007	300 baud
B600	0000010	600 baud
B1200	0000011	1200 baud
B1800	0000012	1800 baud
B2400	0000013	2400 baud
B4800	0000014	4800 baud
B9600	0000015	9600 baud
B19200	0000016	19200 baud
B38400	0000017	38400 baud
EXTA	0000016	External A
EXTB	0000017	External B
CSIZE	0000060	Character size:
CS5	0	5 bits
CS6	0000020	6 bits
CS7	0000040	7 bits
CS8	0000060	8 bits
CSTOPB	0000100	Send two stop bits, otherwise one.
CREAD	0000200	Enable receiver.

PARENB	0000400	Parity enable.
PARODD	0001000	Odd parity, otherwise even.
HUPCL	0002000	Hang up on last close.
CLOCAL	0004000	Local line, otherwise dial-up.
LOBLK	0010000	Block layer output.

The CBAUD bits specify the baud rate. The zero baud, B0, is used to hang up the connection. If B0 is specified, the data-terminal-ready signal will not be asserted. Normally, this will disconnect the line. For any particular hardware, impossible speed changes are ignored.

The CSIZE bits specify the character size in bits for both transmission and reception. This size does not include the parity bit, if any. If CSTOPB is set, two stop bits are used, otherwise one stop bit. For example, at 110 baud, two stops bits are required.

If PARENB is set, parity generation and detection is enabled and a parity bit is added to each character. If parity is enabled, the PARODD flag specifies odd parity if set, otherwise even parity is used.

If CREAD is set, the receiver is enabled; otherwise no characters will be received.

If HUPCL is set, the line will be disconnected when the last process with an open line, closes it or terminates. That is, the data-terminal-ready signal will not be asserted.

If CLOCAL is set, the line is assumed to be a local, direct connection with no modem control; otherwise modem control is assumed.

If LOBLK is set, the output of a job control layer will be blocked when it is not the current layer; otherwise the output generated by that layer will be multiplexed onto the current layer.

The initial hardware control value after open is B300, CS8, CREAD, HUPCL.

The c_lflag field of the argument structure is used by the line discipline to control terminal functions. The basic line discipline (0) provides the following:

ISIG	0000001	Enable signals.
ICANON	0000002	Canonical input (erase and kill processing).
XCASE	0000004	Canonical upper/lower presentation.
ECHO	0000010	Enable echo.
ECHOE	0000020	Echo erase character as BS-SP-BS.

ECHOK	0000040	Echo NL after kill character.
ECHONL	0000100	Echo NL.
NOFLSH	0000200	Disable flush after interrupt or quit.

If ISIG is set, each input character is checked against the special control characters INTR, SWTCH, and QUIT. If an input character matches one of these control characters, the function associated with that character is performed. If ISIG is not set, no checking is done. Thus these special input functions are possible only if ISIG is set. These functions may be disabled individually by changing the value of the control character to an unlikely or impossible value (for example, 0377).

If ICANON is set, canonical processing is enabled. This enables the erase and kill edit functions, and the assembly of input characters into lines delimited by NL, EOF, and EOL. If ICANON is not set, read requests are satisfied directly from the input queue. A read will not be satisfied until at least MIN characters have been received or the timeout value TIME has expired between characters. This allows fast bursts of input to be read efficiently while still allowing single character input. The MIN and TIME values are stored in the position for the EOF and EOL characters, respectively. The time value represents tenths of seconds.

If XCASE is set, and if ICANON is set, an uppercase letter is accepted on input by preceding it with a \ character, and on output is preceded by a \ character. In this mode, the following escape sequences are generated on output and accepted on input:

for:	*use*:
`	\'
|	\!
~	\^
{	\(
}	\)
\	\\

For example, A is input as \a, \n as \\n, and \N as \\\n.

If ECHO is set, characters are echoed as received.

When ICANON is set, the following echo functions are possible. If ECHO and ECHOE are set, the erase character is echoed as ASCII BS SP BS, which will clear the last character from a CRT screen. If ECHOE is set and ECHO is not set, the erase character is echoed as ASCII SP BS. If ECHOK is set, the NL charac-

ter will be echoed after the kill character to emphasize that the line will be deleted. Note that an escape character preceding the erase or kill character removes any special function. If ECHONL is set, the NL character will be echoed even if ECHO is not set. This is useful for terminals set to local echo (so-called half duplex). Unless escaped, the EOF character is not echoed. Because EOT is the default EOF character, this prevents terminals that respond to EOT from hanging up.

If NOFLSH is set, the normal flush of the input and output queues associated with the quit, switch, and interrupt characters will not be done.

The initial line-discipline control value is all bits clear.

The primary ioctl(2) system calls have the form

```
ioctl (fildes, command, arg)
struct termio *arg;
```

The commands using this form are

TCGETA	Get the parameters associated with the terminal and store in the termio structure referenced by *arg*.
TIOCGPGRP	The current terminal process group is placed into the word at the address contained in *arg*.
TIOCSPGRP	The address pointed to by *arg* contains a word, typically a process ID, that becomes the process group for the controlling terminal.
TCSETA	Set the parameters associated with the terminal from the structure referenced by *arg*. The change is immediate.
TCSETAW	Wait for the output to drain before setting the new parameters. This form should be used when changing parameters that will affect output.
TCSETAF	Wait for the output to drain, then flush the input queue and set the new parameters.

Additional `ioctl`(2) calls have the form

```
ioctl (fildes, command, arg)
int arg;
```

The commands using this form are

`TCSBRK`	Wait for the output to drain. If *arg* is 0, then send a break (zero bits for 0.25 seconds).
`TCXONC`	Start/stop control. If *arg* is 0, suspend output; if 1, restart suspended output; if 2, suspend input; if 3, restart suspended input.
`TCFLSH`	If *arg* is 0, flush the input queue; if 1, flush the output queue; if 2, flush both the input and output queues.

The following `ioctl`(2) calls take the form

```
ioctl (fildes, command, 0)
```

They are for modem control; not all devices support all or any of them. If any are supported then `UIOCTTSTAT` is supported. The default is `UIOCNOMODEM/UIOCNOFLOW`. All these are "remembered" when a device is closed and reopened again.

The following are mutually exclusive (on some systems DTR/DCD are named in reverse order: here DCD is the input, DTR the output).

`UIOCMODEM`	Modem control (DTR/DCD) is enabled (DCD is required before a device can be opened, if it is removed the device is "hung up," and on opening DTR is asserted); the default is "on" for `/dev/modem` and `/dev/tty0`.
`UIOCEMODEM`	"European style" modem control. Like `UIOMODEM` except that DTR is not asserted until RI (ring interrupt) is detected.
`UIOCNOMODEM`	No modem control. DTR is still asserted, DCD is ignored, and opens always complete without waiting.
`UIOCDTRFLOW`	The DCD (on some printers this is the DTR line) is used for flow control. It must be asserted before characters can be transmitted; the default is "on" for `/dev/printer` and `/dev/tty1`.

The next two commands are also mutually exclusive. Again, CTS/RTS are sometimes named in reverse order. Here RTS is the output, CTS the input.

`UIOCNOFLOW` Hardware flow control is disabled. RTS is asserted before transmitting (or asserted all the time). CTS is ignored.

`UIOCFLOW` Hardware flow control is enabled. RTS is asserted before transmitting. CTS must be asserted by the other end before transmission can start. (This is required for every character.)

`UIOCTTSTAT` This returns 3 bytes. The first is 1 if `UIOCMODEM` is enabled. The second is 1 if `UIOCDTRFLOW` is enabled and the third is 1 if `UIOCFLOW` is enabled.

The following `ioctl`(2) calls have the form

```
ioctl (fildes, command, pArg)
int *pArg;
```

The commands that use this form are

`FIONREAD` Return the number of characters currently in a terminal's input buffer into the integer pointer **pArg*.

`FIONBIO` If the integer referenced by the pointer *pArg* is 1, then turn on nonblocking IO. If it is 0, turn it off. If nonblocking IO is turned on, then read or write will return without blocking (and return the error `EWOULDBLOCK`) if it is not possible to complete the transfer immediately.

`FIOASYNC` If the integer referenced by the pointer *pArg* is 1, turn on I/O signalling. If it is 0, turn it off. If I/O signalling is turned on then the signal `SIGIO` will be sent whenever input is available to the device. Care should be taken that all the processes in the tty's process group can respond (or ignore) this signal if it is enabled.

Some devices are streams based. In order for them to respond to the `ioctl` calls discussed here, the streams module (line discipline) `line` must be pushed on the stream. For most logged in ter-

minals, this is done by /etc/getty as part of the logging in process. Refer to streams(7) and line_push(3) for more information about how to do this if it is required.

SERIAL MANAGER SUPPORT

The following new ioctl calls were added to support the Serial Manager running under A/UX 2.0. Five of the calls have the form

```
ioctl (fildes, command, 0)
```

The commands using this form are as follows:

TCRESET	Reset the serial line identified by *filedes*.
TCSETDTR	Drive the DTR line high for the serial line identified by *filedes*. This turns DTR on.
TCCLRDTR	Drive the DTR line low for the serial line identified by *filedes*. This turns DTR off.
TCSBRKM	Set break mode for the serial line identified by *filedes*. This starts a line break signal.
TCCBRKM	Clear break mode for the serial line identified by *filedes*. This terminates a line break signal.

Two of the calls have the form

```
ioctl (fildes, command, arg)
int *arg;
```

The commands using this form are as follows:

TCSETSTP	Set the stop character for flow control for the serial line identified by *filedes*. *arg* points to a byte containing the new stop character.
TCSETSTA	Set the start character for flow control for the serial line identified by *filedes*. *arg* points to a byte containing the new start character.

One call has the form

```
ioctl (fildes, command, arg)
struct serstat *arg;
```

where the serstat structure has the following format:

```
struct serstat {
   unsigned long ser_frame;    /*framing errors*/
   unsigned long ser_ovrun;    /*overrun errors*/
   unsigned long ser_parity;   /*parity errors */
   unsigned long ser_cts;      /*CTS signal     */
   unisgned long ser_inflow;   /*input flow     */
                               /*control        */
   unsigned long ser_outflow; /*output flow    */
                               /*control        */
};
```

The command using this form is as follows:

TCGETSTAT — Get status information for the serial identified by *filedes* and store it in the B termio structure referenced by *arg*. The `ser_frame`, `ser_ovrun`, and `ser_parity` members of the `serstat` structure represent the error counts that have been tallied since the last call to `TCGETSTAT`. The `ser_cts` member indicates the current status of the CTS signal. A true value indicates that CTS is on (high); otherwise, CTS is off (low). If the `ser_inflow` member is true, input is currently blocked due to flow control. If the `ser_outflow` member is true, output is currently blocked due to flow control.

4.2 BSD COMPATIBLE FEATURES

Local Special Characters

When job control is active, there is a `ltchars` structure associated with each terminal. Two fields are used, which define characters to stop a process. The other fields are for compatibility with past and future systems.

```
struct ltchars {
   char t_suspc;    /* stop process signal  */
   char t_dsuspc;   /* delayed stop process */
                    /* signal               */
   char t_rprntc;   /* Not used             */
   char t_flushc;   /* Not used             */
   char t_werase;   /* Not used             */
   char t_lnextc;   /* Not used             */
};
```

By default, these characters are disabled (set to −1). Traditionally CONTROL-Z is used for the suspend character and CONTROL-Y for the delayed suspend.

TIOCSLTC — The *arg* parameter to the `ioctl` call (as shown above) is the address of a `ltchars` structure which defines the new local characters.

TIOCGLTC — The *arg* parameter to the `ioctl` is the address of a `ltchars` structure into which the current set of special characters is placed.

Compatibility Modes

An additional mode word is recognized by the BSD compatible TTY driver. It is used to set and clear the job control "tostop" bit. When set, processes running in the background which write on the terminal will be sent a `SIGTTOU` signal. When BSD compatible signals are used, background processes which read from TTY will be sent `SIGTTIN`.

TOSTOP — 0x1–Send `SIGTTOU` for background output.

TIOCSCOMPAT — The *arg* parameter to the `ioctl` call is the address of an integer containing the new value of the compatibility mode word.

TIOCGCOMPAT — The *arg* parameter is the address of an integer variable chosen to receive the new compatibility mode word.

FILES

```
/dev/tty
/dev/tty*
/dev/console
```

SEE ALSO

`stty`(1), `getty`(1M), `fork`(2), `ioctl`(2), `read`(2), `setpgrp`(2), `line_push`(3), `signal`(3), `streams`(7).

NAME

termios — A/UX® POSIX general terminal interface

SYNOPSIS

```
#include <termios.h>
```

DESCRIPTION

Part of the A/UX POSIX environment is a general terminal interface for controlling asynchronous communications ports.

When a terminal file is opened, it normally causes the process to wait until the connection is established. In practice, user programs seldom open these files; getty(1M) opens them, and they become a user's standard input, output, and error files.

The file /dev/tty is, in each process, the control terminal associated with the process group of that process. Programs or shell sequences use it to ensure that their messages appear on the terminal, no matter how output is redirected. Also, programs that demand an output filename accept /dev/tty, so it is not necessary to determine which terminal is being used.

Opening a terminal device causes the process to block until the connection is established. If the O_NONBLOCK flag is set, open(2) returns a file descriptor without waiting for the connection to be established.

A terminal may have a foreground process group associated with it. Certain characters have special functions on input or output. The foreground process group plays a role in the handling of signal-generating characters.

Shells that support job control can allocate the terminal to different jobs, or process groups, by placing related processes in a single process group and associating this process group with the terminal. The associated process group of a terminal may be set or examined by a process in the process group by using tcsetpgrp(3P) and tcgetpgrp(3P).

A terminal may belong to a process as its controlling terminal. Each process of a session that has a controlling terminal has the same controlling terminal. A terminal may be the controlling terminal for at most one session. If the process ID of the calling process is equal to the process group ID and the process has no controlling terminal, the next open of a terminal without a controlling process causes the opened terminal to become the controlling terminal for the process. If a process that is not a session leader

opens a terminal file or the O_NOCTTY flag is used when calling open, the terminal does not become the controlling terminal of the process. If the O_GETCTTY flag is used when calling open, the terminal becomes the controlling terminal of the calling process. When a controlling terminal becomes associated with a session, its foreground process group is set to the process group of the session leader.

The controlling terminal is inherited by a child process during a fork (see fork(2)). A process relinquishes its controlling terminal when it changes its process group by using setsid(2). When a controlling process terminates, the foreground process group of its controlling terminal is set to 0. This allows the terminal to be acquired as a controlling terminal by a new process group of the session leader.

A terminal device associated with a terminal device file may operate in full-duplex mode so that characters may arrive even while output is occurring. Each terminal device file has associated with it an input queue, into which incoming characters are placed by the system before being read by a process. The system imposes a limit, MAX_INPUT, on the number of bytes that may be stored in the input queue. If MAX_INPUT is exceeded, the queue is flushed.

A terminal device file may be in canonical mode or noncanonical mode. The mode of the terminal device file determines the method of input processing.

In canonical-mode input processing, terminal input is processed in units of lines. A line is delimited by a newline (\n) character and an end-of-file (EOF) or end-of-line (EOL) character. This means that a read request is not satisfied until an entire line is typed or a signal is received. Also, no matter how many characters are requested by the read, at most one line is returned. It is not necessary to read a whole line at once; any number of characters, even one, may be requested in a read without losing information. MAX_CANON is the limit on the number of bytes in a line. If this limit is exceeded, the input buffer is flushed. Erase and kill processing occurs during canonical-mode input processing.

In noncanonical-mode input processing, input characters are not assembled into lines, and erase and kill processing does not occur. The values of the special characters MIN and TIME are used to determine how to process the characters received. MIN and TIME

are defined in the c_cc array of special control characters.

MIN represents the minimum number of characters that should be received when the read is satisfied. TIME is a timer of 0.1 second granularity that is used to time out data characterized by short bursts and short-term data transmissions. The four possible combinations for MIN and TIME are as follows:

MIN>0, TIME>0
: In this case TIME serves as an interbyte timer and is activated after the first byte is received. Since it is an interbyte timer, it is reset after a byte is received. When the first byte is received, the interbyte timer is started. If MIN bytes are received before the timer expires, the read is satisfied. If the timer expires before MIN bytes are received, the bytes received to that point are returned to the user. Note that if TIME expires, at least one byte is returned because the timer is not started unless a byte has been received. In this case, the read blocks until the MIN and TIME mechanisms are activated by the receipt of a byte.

MIN>0, TIME=0
: When the value of TIME is 0, the timer plays no role, and only MIN is significant. A pending read is not satisfied until MIN bytes are received. A program that sets TIME to 0 when reading record-based terminal I/O may block indefinitely on a read operation.

MIN=0, TIME>0
: When MIN is 0, TIME no longer represents an interbyte timer. TIME now serves as a read timer that is activated as soon as the read(2) is processed. A read is satisfied as soon as a single byte is received or the read timer expires. Note that if the timer expires, no byte is returned. If the timer does not expire, the only way the read can be satisfied is if a byte is received. In this case, reads do not block indefinitely waiting for a byte; if no byte is received within TIME*0.1 seconds after the read is initiated, the read returns 0 bytes.

MIN=0, TIME=0
: The minimum of either the number of bytes requested or the number of bytes currently available is returned without waiting for more bytes to be received.

Reads are also dependent on the whether the O_NONBLOCK flag is set by the open(2) or fcntl(2) call. If the O_NONBLOCK flag is not set, then a read request blocks until data is available or a signal is received. If the O_NONBLOCK flag is set, then a read completes without blocking in one of three ways:

1. If there is enough data available to satisfy the entire request, the read completes successfully, having read all the requested data, and returns the number of bytes read.
2. If there is not enough data available to satisfy the entire request, the read completes successfully, having read as much data as possible, and returns the number of bytes it was able to read.
3. If there is no data available, the read returns –1, and errno is set to EAGAIN.

Any attempt by a process in a background process group to read from its controlling terminal causes its process group to be sent a SIGTTIN signal unless the reading process is ignoring or blocking SIGTTIN , or the process group of the reading process is orphaned. Then the read(2) returns -1 with errno set to EI0, and no signal is sent. The default action of SIGTTIN is to stop the process to which it is sent.

Any attempt by a process in a background process group to write to its controlling terminal will cause the process group to be sent a SIGTTOU signal unless one of the following special cases apply: If TOSTOP is not set, or if TOSTOP is set and the process is ignoring or blocking SIGTTOU, the process is allowed to write to the terminal, and SIGTTOU is not sent. If TOSTOP is set and the process group of the writing process is orphaned and if the writing process is not ignoring or blocking SIGTTOU, the write(2) returns -1 with errno set to EI0, and no signal is sent.

Routines that need to control terminal characteristics do so by modifying the termios structure for the device. This structure is defined in <termios.h> as follows:

```
struct termios {
    lcflag_t long   c_iflag;
    lcflag_t long   c_oflag;
    lcflag_t long   c_cflag;
    lcflag_t long   c_lflag;
    char            c_line;
```

```
        cc_t char       c_cc[NCCS];
};
```

The `c_iflag` field describes the basic terminal input control. The following flags are defined by POSIX:

IGNBRK	0000001	Ignore break condition.
BRKINT	0000002	Signal interrupt on break.
IGNPAR	0000004	Ignore characters with parity errors.
PARMRK	0000010	Mark parity errors.
INPCK	0000020	Enable input parity check.
ISTRIP	0000040	Strip character.
INLCR	0000100	Map NL to CR on input.
IGNCR	0000200	Ignore CR.
ICRNL	0000400	Map CR to NL on input.
IXON	0002000	Enable start/stop output control.
IXOFF	0010000	Enable start/stop input control.

In addition, A/UX supports the following flags:

ICRNL	0000400	Map CR to NL on input.
IUCLC	0001000	Map uppercase to lowercase on input.
IXANY	0004000	Enable any character to restart output.

If IGNBRK is set, a break condition detected on input is ignored; it is not put on the input queue and therefore not read by any process. If IGNBRK is not set and BRKINT is set, the break condition flushes both the input and output queues, and if the terminal is the controlling terminal of a foreground process group, a SIGINT signal is sent to that foreground process group. If neither IGNBRK nor BRKINT is set, a break condition is read as a single '\0,' or if PARMRK is set, as '\377,' '\0,' '\0.' If IGNPAR is set, a byte with a framing or parity error is ignored.

If PARMRK is set and IGNPAR is not set, a byte with a framing or parity is put on the input queue as the three-character sequence '\377,' '\0,' X, where '\377,' '\0,' is a two-character flag and X is the data of the byte received in error. To avoid ambiguity in this case, if ISTRIP is not set, a valid character of '\377' is given to the application as '\377,' '\377.' If neither PARMRK nor IGNPAR is set, a framing or parity error is put on the input queue as a single character, '\0.'

If INPCK is set, input parity checking is enabled. If INPCK is not set, input parity checking is disabled, allowing output parity generation without input parity errors. Note that whether input parity

checking is enabled or disabled is independent of whether parity detection is enabled or disabled. If parity detection is enabled but input parity checking is disabled, the hardware to which the terminal device file is connected recognizes the parity bit, but the terminal special file does not check whether this bit is set correctly.

If ISTRIP is set, valid input characters are first stripped to 7 bits; otherwise, all 8 bits are processed.

If INLCR is set, a received newline (NL) character is translated into a RETURN character. If IGNCR is set, a received RETURN character is ignored (not read). If ICRNL is set, a received RETURN character is translated in an NL character.

If IUCLC is set, a received uppercase alphabetic character is translated into the corresponding lowercase character.

If IXON is set, start/stop output control is enabled. A received STOP character suspends output and a received START character restarts output. All start/stop characters are not read, but perform flow-control functions. If IXANY is set, any input character restarts output that was suspended.

If IXOFF is set, the system transmits START/STOP characters when the input queue is nearly empty or full.

The initial input control value is all-bits-clear.

The c_oflag field specifies the system treatment of output. POSIX defines the following flag for c_oflag:

OPOST	0000001	Postprocess output.

In addition, A/UX supports the following flags:

OLCUC	0000002	Map lowercase to uppercase on output.
ONLCR	0000004	Map NL to CR-NL on output.
OCRNL	0000010	Map CR to NL on output.
ONOCR	0000020	No CR output at column 0.
ONLRET	0000040	Use NL to perform CR function.
OFILL	0000100	Use fill characters for delay.
OFDEL	0000200	Fill is DEL, else NNUL.
NLDLY	0000400	Select newline delays:
NL0	0	
NL1	0000400	
CRDLY	003000	Select carriage-return delays:
CR0	0	
CR1	0001000	

CR2	0002000	
CR3	0003000	
TABDLY	0014000	Select horizontal tab delays:
TAB0	0	
TAB1	0004000	
TAB2	0010000	
TAB3	0014000	Expand tabs to spaces.
BSDLY	0020000	Select backspace delays:
BS0	0	
BS1	0020000	
VTDLY	0040000	Select vertical tab delays:
VT0	0	
VT1	0040000	
FFDLY	0100000	Select form feed delays:
FF0	0	
FF1	0100000	

If OPOST is set, output characters are postprocessed as indicated by the remaining flags; otherwise, characters are transmitted without change.

If OLCUC is set, a lowercase alphabetic character is transmitted as the corresponding uppercase character. This function is often used in conjunction with IUCLC.

If ONLCR is set, the NL character is transmitted as the CR-NL character pair. If OCRNL is set, the CR character is transmitted as the NL character. If ONOCR is set, no CR character is transmitted when at column 0 (first position). If ONLRET is set, the NL character is assumed to do the carriage-return function, the column pointer are set to 0, and the delays specified for CR are used. Otherwise, the NL character is assumed to do just the line feed function; the column pointer remains unchanged. The column pointer is also set to 0 if the CR character is actually transmitted.

The delay bits specify how long the transmission stops to allow for mechanical or other movement when certain characters are sent to the terminal. In all cases a value of 0 indicates no delay. If OFILL is set, fill characters are transmitted for delay instead of a timed delay. This is useful for high-baud-rate terminals that need only a minimal delay. If OFDEL is set, the fill character is DEL, otherwise NUL.

If a form feed delay or vertical tab delay is specified, it lasts for about 2 seconds.

The newline delay lasts about 0.10 seconds. If ONLRET is set, the carriage-return delays are used instead of the newline delays. If OFILL is set, two fill characters will be transmitted.

The carriage-return delay, type 1, is dependent on the current column position. Type 2 is about 0.10 seconds, and type 3 is about 0.15 seconds. If OFILL is set, delay type 1 transmits two fill characters, and type 2, four fill characters.

The horizontal tab delay, type 1, is dependent on the current column position. Type 2 is about 0.10 seconds. Type 3 specifies that tabs are to be expanded into spaces. If OFILL is set, two fill characters are transmitted for any delay.

The backspace delay lasts about 0.05 seconds. If OFILL is set, one fill character is transmitted.

The actual delays depend on line speed and system load.

The initial output control value is all bits clear.

The c_cflag field describes the hardware control of the terminal. POSIX defines the following flags:

CSIZE	0000060	Character size:
CS5	0	5 bits
CS6	0000020	6 bits
CS7	0000040	7 bits
CS8	0000060	8 bits
CSTOPB	0000100	Send two stop bits, else one.
CREAD	0000200	Enable receiver.
PARENB	0000400	Enable parity.
PARODD	0001000	Odd parity, else even.
HUPCL	0002000	Hang up on last close.
CLOCAL	0004000	Local line, else dial-up.

In addition, A/UX POSIX supports the following flags:

LOBLK	0010000	Block layer output.
CBAUD	0000017	Baud rate:
B0	0	Hang up
B50	0000001	50 baud
B75	0000002	75 baud
B110	0000003	110 baud
B134	0000004	134.5 baud

B150	0000005	150 baud
B200	0000006	200 baud
B300	0000007	300 baud
B600	0000010	600 baud
B1200	0000011	1200 baud
B1800	0000012	1800 baud
B2400	0000013	2400 baud
B4800	0000014	4800 baud
B9600	0000015	9600 baud
B19200	0000016	19200 baud
B38400	0000017	38400 baud
EXTA	0000016	External A
EXTB	0000017	External B

The CBAUD bits specify the baud rate. The zero baud rate, B0, is used to hang up the connection. If B0 is specified, the data-terminal-ready signal is not asserted. Normally, this disconnects the line. For any particular hardware, impossible speed changes are ignored.

The CSIZE bits specify the character size in bits for both transmission and reception. This size does not include the parity bit, if any. If CSTOPB is set, two stop bits are used, otherwise, one stop bit. For example, at 110 baud, two stops bits are required.

If PARENB is set, parity generation and detection is enabled, and a parity bit is added to each character. If parity is enabled, the PARODD flag specifies odd parity if set; otherwise, even parity is used.

If CREAD is set, the receiver is enabled. Otherwise, no characters can be received.

If HUPCL is set, the line is disconnected when the last process with the line open closes it or terminates. That is, the data-terminal-ready signal is not asserted.

If CLOCAL is set, the line is assumed to be a local, direct connection with no modem control. Otherwise, modem control is assumed.

If LOBLK is set, the output of a job control layer is blocked when it is not the current layer. Otherwise, the output generated by that layer is multiplexed onto the current layer.

The initial hardware control value after open is B300, CS8, CREAD, HUPCL.

The c_lflag field of the argument structure is used by the line discipline to control terminal functions. The basic line discipline (0) provides the following:

ISIG	0000001	Enable signals.
ICANON	0000002	Enable canonical input (erase and kill processing).
IEXTEN		Enable extended functions.
ECHO	0000010	Enable echo.
ECHOE	0000020	Echo erase character as BS-SP-BS.
ECHOK	0000040	Echo NL after kill character.
ECHONL	0000100	Echo NL.
NOFLSH	0000200	Disable flush after interrupt or quit.
TOSTOP	0100000	Send SIGTTOU for background output.

In addition, A/UX provides:

XCASE	0000004	Enable canonical uppercase and lowercase presentation.

If ISIG is set, each input character is checked against the special control characters INTR, SUSP, SWTCH, and QUIT. If an input character matches one of these control characters, the function associated with that character is performed. If ISIG is not set, no checking is done. Thus these special input functions are possible only if ISIG is set. These functions may be disabled individually by changing the value of the control character to an unlikely or impossible value, such as 0377.

If ICANON is set, canonical processing is enabled. This enables the erase and kill edit functions as well as the assembly of input characters into lines delimited by NL, EOF, and EOL. If ICANON is not set, read requests are satisfied directly from the input queue. A read is not satisfied until at least MIN characters have been received or the timeout value TIME has expired between characters. This allows fast bursts of input to be read efficiently while still allowing single character input. The MIN and TIME values are stored in the position for the EOF and EOL characters, respectively. The time value represents tenths of seconds.

There are currently no extended functions; therefore, the IEXTEN option has no effect.

If XCASE is set and if ICANON is set, an uppercase letter is accepted on input by preceding it with a \ character and is output preceded by a \ character. In this mode, the following escape sequences are generated on output and accepted on input:

for:	*use*:
`	\'
|	\!
~	\^
{	\(
}	\)
\	\\

For example, A is input as \a, \n as \\n, and \N as \\\n.

If ECHO is set, characters are echoed as received.

When ICANON is set, the following echo functions are possible. If ECHO and ECHOE are set, the erase character is echoed as ASCII BS- SP- BS , which clears the last character from a CRT screen. If ECHOE is set and ECHO is not set, the erase character is echoed as ASCII SP-BS . If ECHOK is set, the NL character is echoed after the kill character to emphasize that the line is deleted. Note that an escape character preceding the erase or kill character removes any special function. If ECHONL is set, the NL character is echoed even if ECHO is not set. This is useful for terminals set to local echo (so-called half duplex). Unless escaped, the EOF character is not echoed. Because EOT is the default EOF character, this prevents terminals that respond to EOT from hanging up.

If NOFLSH is set, the normal flush of the input and output queues associated with the quit, suspend, switch, and interrupt characters is not done.

The initial line-discipline control value is all-bits-clear.

The special characters with their default values and functions are as follows:

INTR CONTROL-C. If the ISIG flag is enabled, generates a SIGINT signal that is sent to all processes in the distinguished process group associated with the terminal.

QUIT ASCII FS. If the ISIG flag is enabled, generates a SIGQUIT signal that is sent to all processes in the distinguished process group associated with the terminal.

ERASE DELETE. If the ICANON flag is set, erases the preceding character. It does not erase beyond the start of a line, as delimited by an NL, EOF, or EOL character.

KILL CONTROL-U. If the ICANON flag is set, deletes the entire line, as delimited by an NL, EOF, or EOL character.

EOF CONTROL-D. If the ICANON flag is set when this character is received, immediately pass to the program all the characters waiting to be read without waiting for a newline, and discard the EOF. If there are no characters waiting, zero characters are passed to the program to indicate an end-of-file condition.

NL ASCII LF. If the ICANON flag is set, acts as the line delimiter (\n). It cannot be changed.

EOL ASCII NUL. If the ICANON flag is set, acts as an additional line delimiter similar to NL .

SUSP CONTROL-Z. If the ISIG flag is set, generates a SIGTSTP signal that is sent to all processes in the distinguished process group associated with the terminal.

STOP CONTROL-S. If IXON or IXOFF flag is set, temporarily suspends output. This character is used on terminals to prevent output from disappearing before it can be read.

START CONTROL-Q. If the IXON or IXOFF flag is set, resumes output that has been suspended by a STOP character.

SWTCH CONTROL-Z. If used by the shell-layering facility, shl, changes the current layer to the control layer.

The START and STOP characters cannot be changed. The values for INTR, QUIT, ERASE, KILL, EOF, EOL, and SUSP can be changed by using tcsetattr(3P). ERASE, KILL, and EOF characters may be escaped by preceding the character with a \; in this case, no special function is performed.

c_line specifies the line discipline number for the terminal. The basic line discipline number is 0; this is currently the only line discipline supported.

Special control characters are defined by the c_cc array in the termios structure. The subscript names and descriptions are as follows:

VEOF	EOF character
VEOL	EOL character
VERASE	ERASE character
VINTR	INTR character
VKILL	KILL character
VQUIT	QUIT character
VSUSP	SUSP character
VMIN	MIN character
VTIME	TIME character

If a modem disconnect is detected by the terminal interface for a controlling terminal and if CLOCAL is not set in the c_cflag field for the terminal, a SIGHUP signal is sent to the controlling process associated with the terminal. Unless other arrangements were made (see signal(3)), this causes the controlling process to terminate. Any subsequent read from the terminal device returns an end-of-line indication until the device is closed. Thus, processes that read a terminal file and test for end-of-file can terminate appropriately after a disconnect. Any subsequent write(2) to the terminal device returns -1, with errno set to EIO, until the device is closed.

The last process to close a terminal device file shall cause any output to be sent to the device and any input to be discarded. If HUPCL is set in the control structure and the communications port supports a disconnect function, the terminal device performs a disconnect.

The following functions are provided for controlling the terminal interface:

cfgetispeed(3P)	Return the input baud rate.
cfgetospeed(3P)	Return the output baud rate.
cfsetispeed(3P)	Set the input baud rate.
cfsetospeed(3P)	Set the output baud rate.
tcdrain(3P)	Wait until all written data is transmitted.
tcflow(3P)	Suspend or restart output or input.
tcflush(3P)	Discard data not transmitted.

tcgetattr(3P)	Get terminal attributes.
tcgetpgrp(3P)	Get distinguished process group ID.
tcsendbreak(3P)	Send a break.
tcsetattr(3P)	Set terminal attributes.
tcsetpgrp(3P)	Set distinguished process group ID.

FILES

/dev/tty

SEE ALSO

cfgetospeed(3P), fcntl(2), getty(1M), open(2), tcdrain(3P), tcgetpgrp(3p), tcgetattr(3P), tcsetpgrp(3P).

NAME

tty — controlling terminal interface

DESCRIPTION

The file /dev/tty is, in each process, the control terminal associated with the process group of that process. Programs or shell sequences use it to ensure that their messages appear on the terminal, no matter how output is redirected. Also, programs that demand an output file name will accept /dev/tty, so you don't have to find out what terminal they are using.

FILES

/dev/tty
/dev/tty*

SEE ALSO

termio(7).

Table of Contents

Section 8: Stand-Alone Commands

NAME

intro — introduction to commands executed from the A/UX Startup shell

DESCRIPTION

The A/UX Startup shell (see StartupShell(8)) works like the Bourne shell, but it runs under the Macintosh Operating System. Although it runs within a Macintosh partition, this shell honors the use of the files and directories from the A/UX root partition. For example, to list the contents of the A/UX directory /etc, the usual command syntax for ls is allowed

```
ls options  /etc
```

The ability to access the files in an A/UX partition can be handy. For example, file system maintenance can be performed without an operable A/UX root file system (use fsck found in the StartupShell). Another advantage is that these commands allow you to exercise control over the boot process through modifications to startup files such as /etc/inittab prior to booting A/UX (use ed under A/UX Startup). Otherwise, such a change would require booting A/UX to make the changes, then rebooting (at least partially) in order to see the changes take effect.

The StartupShell versions of A/UX commands that are available include:

```
cat
chgrp
chmod
chown
cp
date
dd
dp
ed
esch
fsck
fsdb
kconfig
launch
ln
ls
mkdir
mkfs
```

```
mknod
mv
od
pname
rm
stty
```

Most of these commands are documented in Section 1 of *A/UX Command Reference*. However, some of them are unique to the A/UX Startup shell. Notably, those commands that provide valuable administration functions such as `esch` (see `esch`(8) and `autorecovery`(8)) are described in the remaining pages of this section.

NAME

autorecovery — file-system repair procedure

DESCRIPTION

Autorecovery refers to the facilities that check and repair A/UX file systems. It includes a variety of tools, some of which run under the A/UX Startup application, and some of which run under A/UX. Autorecovery depends on the existence of one or more parallel file systems containing copies of critical system files. These file systems are maintained on an incremental basis using the appropriate autorecovery programs. The actual checking and correction tasks are performed within the Macintosh portion of the boot sequence and are invoked manually as desired from within the A/UX Startup application. The checking and correction portions of autorecovery can also run automatically upon booting, if the environment has been set up properly.

autorecovery is also the name of an A/UX StartupShell variable that contains the autorecovery command string. This command string is executed whenever you boot A/UX (see StartupShell(8)). The command string has been initially set to echo no autorecovery as an indication that autorecovery is not being run.

To request autorecovery manually, run the esch command directly from the StartupShell. An alternative is to set autorecovery so that the desired form of the esch command is run automatically upon booting.

The A/UX programs that help support the proper functioning of autorecovery are those listed next with the "1M" designation.

SEE ALSO

escher(1M), eu(1M), eupdate(1M), esch(8), StartupShell(8).

NAME

boot — startup procedures

DESCRIPTION

The system is started in a two-stage process. First, the Macintosh operating system is brought up. If the Startup application has been configured properly, the A/UX Startup shell (see StartupShell(8)) will be started automatically. The Startup shell either launches A/UX automatically or prints a Startup shell prompt. If you get the prompt, typing boot causes the boot sequence to be initiated.

SEE ALSO

StartupShell(8), launch(8).
A/UX Installation Guide.

NAME

esch — validate and repair file systems from the A/UX StartupShell

SYNOPSIS

esch [-b] [-c*cluster-number*] [-f] [-v]

DESCRIPTION

esch attempts to ensure that a minimal A/UX file system exists for a multiuser boot. When possible, it corrects bad blocks, repairs file system inconsistencies, and replaces corrupt or missing files. esch is intended to be run when there is reason to suspect that the A/UX file systems have been damaged.

Flag options to esch are

-b Bypasses bad block checking functions.

-c *cluster-number*
Allows the user to specify the autorecovery cluster number.

-f Does not perform fsck (see fsck(1M)).

-v Reports any corrective measures taken.

esch must be run in the A/UX Startup environment before the A/UX system is booted. When the system is powered on or A/UX is rebooted, and the boot dialog is canceled by the user, the A/UX Startup shell (see StartupShell(8)) prompt appears and the esch command line may be entered.

Within the A/UX StartupShell, esch may be set to run automatically with each boot process by assigning the desired esch command line to the StartupShell variable autorecovery.

A hard disk may be divided into partitions. Each partition contains one file system (see fs(4)). Information on all the partitions on a disk is kept in the disk partition map (see dpme(4)) for that disk. One of the fields in a dpme is the cluster number. This is used to identify the group of partitions esch should use.

esch requires that all partitions reside on one disk. esch reads the cluster number from nvram (see nvram(7)) and locates all the partitions in that cluster. A cluster must contain a root partition, a swap partition, and at least one autorecovery partition. A cluster may also contain a partition known to esch as the usr partition. There may be multiple autorecovery partitions in a cluster.

Autorecovery file systems contain copies of files that are necessary for a minimal multiuser A/UX system. If new versions of commands, programs, or files are installed on the system, the autorecovery file systems should be updated using eu(1M) or escher(1M).

esch will check each file system for bad blocks. This is done by verifying that each block can be read. If possible, any blocks that cannot be read will be spared by the hardware or replaced with alternate blocks (see altblk(4)) by the software. If neither of these is successful, the block number is added to a list of blocks that is passed to fsck. fsck will add these blocks to a file associated with inode one; this has the effect of removing the blocks from the file system. Checking for bad blocks is a time-consuming process. This phase of autorecovery may be omitted using the -b flag option. It is advisable to occasionally run esch without the -b flag option to be sure any bad blocks have been dealt with in the appropriate manner.

esch will then run fsck on each file system. This invocation of fsck uses the -y flag option. All questionable files will be removed from the file system. If fsck should fail or if the superblock is unreadable, a mkfs (see mkfs(1M)) will be performed on the file system.

After the file systems have been checked for consistency, esch will enter the file check merge phase. The configuration master list (see cml(4)) is a list of files required for a multiuser A/UX system. This file gives rules about the attributes of each required file. The file check merge phase of esch will check each file in the cml for conformity to the specified file attributes (size, version, check sum, permissions, and so forth). If a file does not conform to these rules, esch will attempt to replace it with a copy from an autorecovery file system. If the file in question is found on an autorecovery file system, it must conform to the cml rules or it will not be placed on the root or usr file system.

FILES

/etc/eschatology/init2files

SEE ALSO

escher(1M), eu(1M), eupdate(1M), fsck(1M), mkfs(1M), altblk(4), cml(4), dpme(4), fs(4), autorecovery(8), StartupShell(8).
"System Startup and Shutdown" in *A/UX Local System Adminis-*

tration.

WARNINGS

esch must *never* be interrupted! Do not power off the system or push the reset button while esch is running. If esch is interrupted, major file system damage may result or entire file systems may be destroyed.

esch will attempt to replace files on two file systems only. These are the root file system and a file system that is intended to be mounted on /usr. Any other file systems will be ignored by esch.

The superblock of a file system contains information describing the file system. If esch is unable to read the superblock of a file system, or if the superblock has a bad magic number, the file system is not usable and an mkfs will be performed on the file system. Everything on the file system will be removed! All user files will be gone and cannot be restored, since the only files esch restores will be those listed in the cml.

If a file system should become full while esch is copying a replacement file to it, esch will attempt to free up space by deleting files from /lost+found, /tmp, /usr/lost+found, or /usr/tmp (depending upon whether the file system is root or usr). Subdirectories and their contents will also be removed from these directories. If the directories have been emptied and there is still no room to copy required files, esch will terminate with an error.

NAME

launch — launch an A/UX kernel from the A/UX Startup environment

SYNOPSIS

launch [-a] [-d] [-f] [-m] [-r] [-v] [-s] [*pathname*]

launch [-n] [-d] [-f] [-m] [-r] [-v] [-s] [*pathname*]

DESCRIPTION

launch loads an A/UX kernel into memory and transfers control to the kernel. launch can only be run from the A/UX Startup application shell (see StartupShell(8)). As launch transfers control to the kernel, it passes a SCSI ID, a logical unit number, and slice zero as parameters. The kernel uses these parameters to locate the root file system (ROOTDEV).

When no *pathname* is specified, launch uses the filename on the first line of the ASCII file /nextunix. The specified kernel (from the command line or from /nextunix) is then checked for an autoconfiguration match.

autoconfig(1M) is run when a software module is present in the kernel, but the required hardware is missing. (Note that autoconfig will NOT be run when hardware is present and the software is missing.)

If autoconfig needs to be run, then the kernel newunix is launched instead and a flag is set indicating that autoconfig is needed.

The format of *pathname* can vary. If you wish to include a device specification the format is:

(device-spec) path

device-spec is described in detail in StartupShell(8). It consists of three comma-separated numbers enclosed in parentheses. The first number is the SCSI ID for a disk, the second is the logical unit (usually zero), and the final number is a slice number. To illustrate, the following command line launches the sunix kernel located within the file system in slice 2 of the disk that is assigned SCSI ID 1:

```
launch (1,0,2)/src/sys/psu/sunix
```

In this example, the kernel would use slice 0 of the disk with SCSI ID 1 as the location for the root partition. So even though the kernel is loaded from a file system in slice 2, the root file system is

still presumed to be in slice zero of the same disk. When the device specification is not provided the values passed to the kernel for the root disk device are taken from the ROOT shell variable. This shell variable is changed using the menu options of the A/UX Startup application shell (see StartupShell(8)). Once set, its value is retained even when the system is shut down.

To allow the kernel to reside in a different location from the root disk device, you can use the -r flag option while setting the ROOT shell variable as desired for the location of the root disk device. The parameters for SCSI ID and logical disk number supplied in the launch command line will be used to locate the kernel, but will not be passed to the kernel as the parameters identifying the root disk device. Instead the parameters passed to identify the root disk device (SCSI ID and logical unit number) will be those stored in the ROOT shell variable.

FLAG OPTIONS

The following flag options are interpreted by launch:

-a Always run autoconfig. That is, run autoconfig even if the kernel appears to match the hardware. Launching newunix turns this option on automatically.

-d Print debugging output. Displays the contents of the auto_data structure. Obscure to anyone not familiar with the kernel.

-f Forces all floppy disks to be ejected and waits for the insertion of a floppy containing an A/UX file system.

Note: No checks are made to ensure the inserted floppy actually has an A/UX file system on it.)

-m The kernel file is on a Macintosh file system.

-n Never run autoconfig. That is, don't run autoconfig even if the kernel and the hardware are mismatched.

-r Use the root partition on the device specified by the $ROOT variable, rather than the device specified by the kernel pathname.

-v Selects a more verbose way of booting the system. The progress bar is not displayed during the process of booting. Instead of routing boot messages to the normally hidden A/UX console window, they are routed directly to a solitary console emulator window. The console emulator window disappears

if the boot reaches multiuser mode successfully, at which time the Macintosh login dialog box appears to replace the console emulator window.

`-s` Load a symbol table along with the kernel, so that a kernel containing the debugger driver module can offer text descriptions of execution addresses when it is activated.

EXAMPLES

`launch`

Launches the default kernel on the current root disk device.

`launch /unix`

Launches the A/UX kernel located in `/unix` on the current root disk device.

FILES

`/nextunix`	Contains name of A/UX kernel to launch
`/newunix`	A/UX kernel to launch if autoconfiguration is needed
`/unix`	The usual name of the current kernel

SEE ALSO

`kconfig`(1M), `StartupShell`(8).

WARNINGS

`launch` will not start a kernel if a PMMU (MC68851) is not present.

NAME

StartupShell — a command interpreter accessible from within the A/UX Startup application

SYNOPSIS

StartupShell

DESCRIPTION

A/UX Startup is a Macintosh® program that can read and execute programs that have been compiled with a special set of libraries under the A/UX® operating system. A/UX Startup provides a command language that executes commands typed at the console or chosen from menus. The language is similar to but simpler than sh(1), csh(1), and ksh(1). Programs run by this shell must reside in a Macintosh file system.

Input

Keyboard input and program output is displayed in a permanently displayed window—the shell window. A subset of the ordinary A/UX terminal interface is supported; specifically, the *erase*, *kill*, *end-of-file*, and *end-of-line* signals; input flags for remapping characters; output flags not associated with delays; and local flags not associated with signals. The control flags are not supported. The *end-of-line* character should not be changed.

The following characters have a special meaning to the shell and help delimit words unless escaped:

< > = *newline space tab*

To suppress their special interpretation, precede any of these characters with a backslash (\). This process is also called an escape, and the backslash is called an escape character when used for this purpose. An escaped *newline* is treated the same as a blank character. All special characters enclosed between a pair of single quotation marks (' '), except another single quotation marks, are treated as if they were escaped. Most special characters enclosed within double quotation marks (" ") are treated as if they were escaped, except the backslash (\), comma (,), and dollar sign ($).

Each line of input is considered to be a single command. A command is broken into words at blanks (a blank is a space or a tab) except when the blanks are escaped.

Text enclosed within double quotation marks that contains references to variables is processed so that variables are replaced by their present values. References to variables enclosed within sin-

gle quotation marks are not replaced with their present value.

A backslash (\) in the first word of a command causes automatic substitution to be disabled for that command.

The A/UX Startup shell prompts with the value of the PS1 variable before reading a command.

Shell Variables

The shell allows certain words to be used as variables. The name of a variable is a sequence of letters, digits, or underscores beginning with a letter or underscore. The exception is the exit status variable (?).

Variables may be assigned values by entering:

name=value

where *value* is a single word, or by entering:

name="multiple-word-value"

where *multiple-word-value* is several words. A dollar sign ($) followed by a variable name is substituted with the present value of a variable. Alternately, you can enclose the variable name in curly braces:

${*name*}

The braces are required only when the name is followed by a letter, digit, or underscore that is not part of the variable name. After variable substitution, the results of substitution are scanned for internal field separator characters (space and tab) and split into distinct arguments where such characters are found. Explicit null arguments ("" or '') are retained. Implicit null arguments (those resulting from variables that have no values) are removed.

A text string may be intrepreted as a variable name without the dollar sign (or braces) if it is the first word of a command line and it is the name of an "automatic" variable. Each time a command line is entered, the first word is checked for matches with any of automatic variables already defined. If there is a match then the value is substituted. The process is not repeated on the replacement.

Whereas normal variables are created when they are assigned values, automatic variables must be declared as such (see auto in the section "Built-in Commands"). The variables autorecovery and autolaunch are examples of automatic

variables.

The following variables are built into the A/UX Startup shell:

PATH	Specify the search path for commands (see the section "Program Execution Environment" below).
PS1	Specify the prompt string, by default "startup# ."
TZ	Specify the A/UX time-zone variable (see environ(5)).
HOME	Specify the default argument (home directory) for the cd and chdir commands.
ROOT	Specify the default argument (home root) for the chroot command.
autorecovery	Specify the command string for verifying the integrity of the file system.
autolaunch	Specify the command string for the kernel launch program (see below).
cwd	Specify the current working directory of the shell.
cwroot	Specify the current working root of the shell.
?	Specify the exit status returned by the last command executed.

The shell gives initial values to all the above variables. The values persist across invocations of A/UX Startup even if the variable is not exported (see export in the section "Built-in Commands"). There is a limit on the number of variables that can exist. The limit is normally 20, but can be changed by modifying a resource.

Program Execution Environment

The shell interprets each line of input as a single command. The first word in the command line is interpreted as the name of the command to be executed.

If a command name matches one of the special commands built into the shell (see the section "Built-in Commands"), it is executed in the shell process. If the command name does not match a special command, a search for a program of the same name occurs in the search paths stored in the PATH variable.

The command name is passed as argument 0 to the executing program. Most of the words following the first word in the command line are passed as arguments to the program invoked. The exceptions are words in positions that are subject to special interpretation because of nearby special characters (see the preceding section "Input").

You can define multiple search paths within the PATH variable by separating them with a vertical bar (|). The default search path is

```
(mac):|(mac):bin:
```

This specifies the same folder as A/UX Startup and a Macintosh folder bin, within the same folder as startup, in that order.

If you enter a command that includes a pathname, as signified by a leading / or (, the search paths in PATH are not used. Otherwise, each directory in the search paths is searched for an executable file.

The current "environment" (see environ(5)) is also passed to each program when invoked. The environment consists of a list of variable names and variable values. The shell maintains the environment in several ways. On invocation, the shell reads the environment from a Macintosh resource. Using the built-in commands, you can modify the values of any of these variables or create new variables. Unless variables are exported (or re-exported) with export, neither their names nor their current values are placed in the environment. Once a variable has been placed in the environment, you can use the unexport or unset, command to remove it.

The environment used by an executing program is thus composed of any unmodified name-value pairs originally read by the shell, minus any pairs removed by unexport and unset, plus any new assignments (or reassignments) that have been added through export commands.

Redirection

The input and the output of a command may be redirected (detached from the console) using specially interpreted characters. The process is called redirection and the special characters are called redirection symbols when used for this purpose. Redirection symbols can also introduce the names of files that are used as the source of input or the destination of output. These redirection symbols and their associated filenames are not passed as argu-

ments to the invoked program. Each of the constructions affects processing as described in the following list:

<word	Use file *word* as standard input (file descriptor 0).
>word	Use file *word* as standard output (file descriptor 1). If the file does not exist, it is created; otherwise, it is truncated to a length of 0.
>>word	Use file *word* as standard output. If the file exists, output is appended to it (by first seeking to the end-of-file); otherwise, the file is created.
<&digit	Use the file associated with file descriptor *digit* as standard input. Similarly, *>&digit* uses the file associated with the file descriptor *digit* as the standard output.
<&-	The standard input is closed. Similarly, standard output is closed by using >&-.

If any of the above are preceded by a digit, the file descriptor is specified by the digit (instead of the default 0 or 1). For example:

```
...2>&1
```

associates file descriptor 2 with the file currently associated with file descriptor 1.

The order in which redirections are specified is significant. The shell evaluates redirections from left to right. For example,

```
... 1>word 2>&1
```

first associates file descriptor 1 with file `word`. It associates file descriptor 2 with the file associated with file descriptor 1 (`word`). If the order of redirections were reversed, file descriptor 2 would be associated with the terminal (assuming file descriptor 1 had been) and file descriptor 1 would be associated with file `word`.

The environment passed to an executing program contains the same file descriptors as the shell, except those modified by input and output redirections on the command line.

Built-in Commands

Built-in commands do not invoke corresponding programs that reside in a file system. Rather, the shell interprets such commands and their arguments all by itself. For these built-in commands, input and output redirection is permitted. Without redirection, the

console is the default output location for standard output and error output. More precisely, file descriptor 1 is the default output location, and error output is permanently directed to the console, bypassing `stderr` (file descriptor 2).

`auto` [*name*]
: Set each of the named variables to be automatic. When entered at the start of a command line, the name of the automatic variable is replaced with its value. To avoid substitution when entering the name of an automatic variable at the start of a command line, precede (escape) the first word with a backslash (\). If entered without any arguments, `auto` displays a list of all automatic variables and their values.

`autolaunch` [*arg*]...
: Execute the command string that has been assigned to the `autolaunch` variable. It cannot be unset or converted into a normal variable, so it consistently behaves like a built-in function. When entered, `autolaunch` executes whatever has been assigned to it. The assignment `autolaunch=launch` causes the startup program named `launch`(8) to be executed when `autolaunch` is entered.

`autorecovery` [*arg*...]
: Executes the command string that has been assigned to the `autorecovery` variable. It cannot be unset or converted into a normal variable, so it consistently behaves like a built-in function. When entered, `autorecovery` executes whatever has been assigned to it.

`boot`
: Execute the `autorecovery` and `autolaunch` command strings. If the `autorecovery` command string fails (`?` is set to a nonzero exit value), `autolaunch` is not started. This command hides the shell window and shows you a progress bar dialog box.

`cd` [*path*]
`chdir` [*path*]
: Change the current directory to that specified by *path.* If no argument is specified, the value of `HOME` is used as the specification for *path.*

`chroot` [*path*]
: Change the current root to that specified by *path.* If no argument is specified, the value of `ROOT` is used as the

specification for *path*.

`default`
Display the default SCSI ID which contains (SCSI ID, 0, 0).

`echo` [*args*]
Display *args* after (unescaped) command and variable substiutions take place.

`eject` [*drive*]
Eject the disk in the enumerated disk drive. The drive number is either 0 or 1.

`exit`
Quit A/UX Startup. (Same as choosing the Quit from the File menu.)

`export` [*names*]
Export each of the named variables so that the variable and its present value become a part of the program execution environment. The present value is saved between invocations of `startup`. If no names are specified, `export` lists all exported variables along with their present values.

`help` [*command*]
Display the help associated with *command*.

`powerdown`
Shut the system power off (same as choosing the Shut Down item from the Execute menu).

`pwd`
Print the current working directory, which is also the value of the variable `cwd`.

`readonly` [*names*]
Set each of the named variables to be read-only. Read-only variables can't be reset through reassignments. However, the variable can be unset and then redefined. If no names are specified, `readonly` displays all the read-only variables along with their values.

`restart`
Restart the system (same as choosing the Restart item from the Execute menu).

`set`
Display all variables and their values.

`shutdown`
Shut down the machine (same as the choosing the Shut Down item from the Execute menu).

`unauto` [*names*]
Remove the automatic attribute from each of the named variables.

`unexport` [*names*]
Remove the `export` attribute from each of the named variables.

`unset` [*names*]
Remove each of the named variables. None of the built-in shell variables can be unset.

Menus

What follows are descriptions of the effect of each menu item. If the menu item presents a dialog box, then each field of the dialog box is also described.

Apple Menu

About A/UX Startup...
Display a window with some information about A/UX Startup, including the use of the `help` function to obtain help information.

Help
Print the default help message in the shell window. The message contains enough information for users to get more help by using the `help` function.

Desk Accessories
Display a menu of the desk accessories currently installed on the system file located in the Macintosh file system from which the system was started. (Desk accessories stored in an A/UX file system won't be available until A/UX is running.)

File Menu

Close
Close the currently active window. However, the shell window cannot be closed.

Quit
Quit A/UX Startup (same as the `exit` command).

Edit Menu

`Undo`
`Cut`
`Copy`
`Paste`
`Clear`

Except for `Copy`, these menu items are present only for use with desk accessories. The shell window only allows Copy, when a range of text is selected.

Execute Menu

Boot

Perform the `autorecovery` and `autolaunch` command strings (same as the `boot` command).

AutoRecovery

Perform the command string that has been assigned to `autorecovery`. Identical to the `autorecovery` command with no arguments.

AutoLaunch

Perform the command string that has been assigned to `autolaunch`. (same as the `autolaunch` command with no arguments except that the progress bar dialog box is shown).

Kill Halt the currently running startup application. COMMAND-PERIOD and COMMAND-K are keyboard shortcuts for this.

Restart

Restart the machine (same as the `restart` command).

Shut Down

Shut the system off (same as the `shutdown` command).

Preferences Menu

Booting ...

Present a dialog box that allows you to set various startup parameters associated with the `boot` command (and the Boot menu item in the Execute menu).

Eject disks on Launch

When this check box is checked, all floppy disks are automatically ejected at the time the A/UX kernel is launched. This value is stored in a Macintosh resource.

Automatically Boot at startup

When this check box is checked, the shell automatically runs the `boot` command when launched, causing the A/UX ker-

nel to boot as a part of the launching of A/UX Startup. This value is stored in a Macintosh resource.

AutoRecovery Command
: This text box displays the value of the built-in `autorecovery` variable. This value can be changed by selecting the text box and editing it.

AutoLaunch Command
: This field displays the value of the built in `autolaunch` variable. The value can be changed by selecting this text box and editing the text displayed.

General ...
: Present a dialog box containing miscellaneous items the user may want to change.

RootDirectory
: This text box displays the value of the built-in `ROOT` variable. The value can be changed by selecting the text and editing it.

Home Directory
: This text box displays the value of the built-in `HOME` variable. The value can be changed by selecting the text and editing it.

Cluster Number
: This text box displays the value of the autorecovery cluster number. Refer to `autorecovery`(8) for an explanation of what this number does. The value is stored in nonvolatile RAM (see `nvram`(7)).

Devices, Partitions, and Pathnames

In the A/UX environment, you access multiple file systems by mounting file systems (block device files) on accessible directories. The A/UX Startup environment does not support `mount`. Instead, the A/UX pathname syntax has been extended with an optional prefixed device specification. A device specification has either the form (S,D,P); where S, D, and P are integers identifying a SCSI ID, disk (or logical unit), and partition respectively, or the special prefix "(default)." The prefix (default) refers to the disk containing the version of A/UX Startup that is currently running.

When no device specification is used, the path is integrated relative to the current directory or current root (as the path is relative or absolute). `chroot` always changes the current directory to be the new root.

Character and block device files are not supported, because they depend on a mapping between device major numbers and devices, which is specific to the A/UX kernel. However, with respect to the following special files, the `open` system call (see `open`(2)) does cause fake device files to be opened in raw (character) mode, but their inodes are outside the normal file system space. All other device files cannot be opened from the A/UX Startup environment. These device files are:

```
/dev/console
/dev/syscon
/dev/systty
/dev/dsk/cSdDsP
/dev/rdsk/cSdDsP
/dev/null
/dev/floppy[0,1]
/dev/rfloppy[0,1]
```

where *S*, *D*, and *P* are integers corresponding to those in a device specification (SCSI ID, device zero, and slice number, respectively).

Disk partitions are assigned slice number. The slice numbers are restricted to the range 0-31. Essentially the slice numbers act as a user-controlled cache of partitions. Three partitions are assigned slice numbers automatically: the root partition to 0, the swap partition to 1, and the usr partition to 2. The way root, swap, and usr partitions are recognized depends on the `autorecovery` cluster number and the block 0 blocks (`bzb`) in the disk-partition-map entries (`dpme`). Slice number 31 always refers to the entire disk. You may also explicitly associate partitions and slice numbers by using `pname`. See `gd`(7), `autorecovery`(8), `dpme`(4), `bzb`(4), and `pname`(1M) for more details.

The following is an example of a device specification and pathname:

```
(1,0,2)/include/sys/param.h
```

This denotes the file `param.h` in the directory `/sys` that is in the directory `/include` that is in the root directory of the file system located in the partition associated with slice 2 of the device with SCSI ID 1.

Programs

The following programs have been converted to run under the A/UX Startup shell:

`cat`	Concatenate and print files.
`chgrp`	Change the group.
`chmod`	Change the mode.
`chown`	Change the owner.
`cp`	Copy files.
`cpio`	Copy file archives in and out.
`date`	Print and set the date..
`dd`	Convert and copy a file.
`dp`	Perform disk partitioning.
`ed`	Edit text.
`esch`	Run the autorecovery program.
`fsck`	Check and interactively repair the file system.
`fsdb`	Debug a SVFS or a UFS file system.
`kconfig`	Change a kernel's variables for tuning.
`launch`	Launch an A/UX kernel.
`ln`	Make links.
`ls`	List the contents of directory.
`mkdir`	Make directories.
`mkfs`	Construct a System V file system.
`mknod`	Build a device file.
`mv`	Move or rename files.
`newfs`	Construct a UFS file system.
`od`	Perform an octal dump.
`pname`	Associate named partitions with device nodes.
`read-disk`	Simplified disk reader for install.
`rm`	Remove files or directories.
`stty`	Set the options for a terminal.
`tar`	File archiver.

Except for `launch`, `esch`, and `read-disk`, each command is an A/UX command rewritten for the A/UX Startup shell environment. For more information about the basically equivalent commands, refer to the other sections of this manual and Section 1 of

the *A/UX Command Reference*.

Variables

None of the built-in variables may be unset or have their attributes changed. Between invocations of A/UX Startup, any reassignments made for built-in variables will persist.

Name	Attribute	Default Value	Comment
?		0	Not assignable
autorecovery	auto	echo no autorecovery	
cwd		/	Not assignable
cwroot		(default)/	Not assignable
HOME	export	/	
autolaunch	auto	launch	
PATH	export	(mac)sl(mac):bin:	
PS1		startup#	
ROOT	export	(default)/	
TZ	export	PST8PDT	

Macintosh Resources

There are several resources in the resource fork of A/UX Startup that may be of interest. All nonstandard resources have associated ResEdit template (TMPL) resources so that they can be edited. One should not edit anything other than the STRL/config resource and the SASH/variables resource. (SASH stands for Startup Application Shell.)

Name	Type	ID	Description
version	SASH	0	A standard version string.
state	SASH	1	Miscellaneous state variables, use SASH template.
variables	SASH	2	Saved variables, exported and built-in, use VARL template.
help	SASH	3	All the help text, use WSTR template.
*	ERRL	*	Connects internal error numbers to problem description and action strings, use ERRL template.
*	STRL	*	Lists of strings identified by numeric tags, use STRL template.
config	STRL	134	A list of strings which are in the following order: font name, font size, maximum number of variables, and kilobytes of memory for running startup programs.

WARNINGS

When using this shell, you are effectively running as superuser, so there is no permission checking. Everything is accessible and very few actions are disallowed.

BUGS

The fake character device files (`/dev/`*rxxxx*) are a bad idea and should be changed. Being required to attach partitions to slices is also bad. Backspacing over tab characters looks wrong on the screen.

Running the `ls` command on `/dev` reports erroneous information for the device files mentioned above. However, you can see the correct information by prefixing a device specification to the directory.

SEE ALSO

dp(1M), pname(1M), bzb(4), dpme(4), environ(5), gd(7), autorecovery(8), launch(8).
A/UX Command Reference.

THE APPLE PUBLISHING SYSTEM

This Apple manual was written, edited, and composed on a desktop publishing system using Apple Macintosh® computers and troff running on A/UX. Proof and final pages were created on Apple LaserWriter® printers. POSTSCRIPT®, the page-description language for the LaserWriter, was developed by Adobe Systems Incorporated.

Text type and display type are Times and Helvetica. Bullets are ITC Zapf Dingbats®. Some elements, such as program listings, are set in Apple Courier.

Writers: J. Eric Akin, Mike Elola, George Towner, and Kathy Wallace

Editor: George Truett

Production Supervisor: Josephine Manuele

Acknowledgments: Lori Falls and Michael Hinkson

Special thanks to Lorraine Aochi, Vicki Brown, Sharon Everson, Pete Ferrante, Kristi Fredrickson, Don Gentner, Tim Monroe, Dave Payne, Henry Seltzer, and John Sovereign

030-0779