ALTOS

586/986

Introduction To XENIX
Appendices

Altos 586/986 Computer Systems
Introduction to XENIX Appendices

ACKNOWLEDGEMENTS

ALTOS is a registered trademark of Altos Computer Systems.

UNIX is a trademark of Bell Laboratories.

XENIX is a trademark of Microsoft, Incorporated and is a 16-bit microcomputer implementation of the UNIX operating system.

WorkNet is a trademark of Altos Computer Systems.

The File Transfer Program for MP/M is copyrighted by the Balcones Computer Corporation.

Change Package to Incorporate the Altos 486 Features into the Introduction to Xenix — Appendices

690-16270-001

Contents

Xenix Commands A

CONTENTS

This appendix, written for programmers, describes XENIX operating system commands that you can use from the XENIX shell or from the Business Shell (precede the command with an !). Commands are listed in alphabetical order.

When you enter a command, use the following format:

```
$ command [options]
```

Options are listed for each command.

Name

ascii – Map of the ASCII character set.

Description

Ascii is a map of the ASCII character set. It lists both octal and hexadecimal equivalents of each character. It contains:

000 nul	001 soh	002 stx	003 etx	004 eot	005 enq	006 ack	007 bel
010 bs	011 ht	012 nl	013 vt	014 np	015 cr	016 so	017 si
020 dle	021 dc1	022 dc2	023 dc3	024 dc4	025 nak	026 syn	027 etb
030 can	031 em	032 sub	033 esc	034 fs	035 gs	036 rs	037 us
040 sp	041 !	042 "	043 #	044 $	045 %	046 &	047 '
050 (	051)	052 *	053 +	054 ,	055 -	056 .	057 /
060 0	061 1	062 2	063 3	064 4	065 5	066 6	067 7
070 8	071 9	072 :	073 ;	074 <	075 =	076 >	077 ?
100 @	101 A	102 B	103 C	104 D	105 E	106 F	107 G
110 H	111 I	112 J	113 K	114 L	115 M	116 N	117 O
120 P	121 Q	122 R	123 S	124 T	125 U	126 V	127 W
130 X	131 Y	132 Z	133 [	134 \	135]	136 ^	137 _
140 `	141 a	142 b	143 c	144 d	145 e	146 f	147 g
150 h	151 i	152 j	153 k	154 l	155 m	156 n	157 o
160 p	161 q	162 r	163 s	164 t	165 u	166 v	167 w
170 x	171 y	172 z	173 {	174 |	175 }	176 ~	177 del

00 nul	01 soh	02 stx	03 etx	04 eot	05 enq	06 ack	07 bel
08 bs	09 ht	0a nl	0b vt	0c np	0d cr	0e so	0f si
10 dle	11 dc1	12 dc2	13 dc3	14 dc4	15 nak	16 syn	17 etb
18 can	19 em	1a sub	1b esc	1c fs	1d gs	1e rs	1f us
20 sp	21 !	22 "	23 #	24 $	25 %	26 &	27 '
28 (	29)	2a *	2b +	2c ,	2d -	2e .	2f /
30 0	31 1	32 2	33 3	34 4	35 5	36 6	37 7
38 8	39 9	3a :	3b ;	3c <	3d =	3e >	3f ?
40 @	41 A	42 B	43 C	44 D	45 E	46 F	47 G
48 H	49 I	4a J	4b K	4c L	4d M	4e N	4f O
50 P	51 Q	52 R	53 S	54 T	55 U	56 V	57 W
58 X	59 Y	5a Z	5b [	5c \	5d]	5e ^	5f _
60 `	61 a	62 b	63 c	64 d	65 e	66 f	67 g
68 h	69 i	6a j	6b k	6c l	6d m	6e n	6f o
70 p	71 q	72 r	73 s	74 t	75 u	76 v	77 w
78 x	79 y	7a z	7b {	7c |	7d }	7e ~	7f del

Files

/usr/pub/ascii

Name

asktime - Prompts for the correct time of day.

Syntax

/etc/asktime

Description

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

```
yymmdd
hhmm
```

Here yy is the last 2 digits of the year number; the first mm is the month number; dd is the day number in the month; hh is the hour number (24-hour system); the second mm is the minute number.

Examples

This example sets the new time, date, and year to "9:23 January 1, 1983".

```
I think it's Wed Nov 3 14:36:23 1982
Enter date (yymmdd) or press RETURN if ok:  830101
Enter time (hhmm) or press RETURN if ok:  0923
```

Diagnostics

If you enter an illegal time, asktime prompts with:

```
Try again:
```

Notes

Asktime is normally performed automatically by the system startup file /.profile immediately after the system is booted; however, it may be executed at any time. The command is privileged, and can only be executed by the superuser.

Name

awk - Invokes a pattern processing editor

Syntax

awk [-Fc] [-f file] [prog] [file] ...

Description

Awk scans each input file for lines that match any of a set of patterns specified in prog. With each pattern in program there can be an associated action that will be performed when a line of a file matches the pattern. The set of patterns may appear literally as prog, or in a file specified as -f file. The prog string should be enclosed in single quotation marks (') to protect it from the shell.

Files are read in order; if there are no files, the standard input is read. The file name '-' means the standard input. Each line is matched against the pattern portion of every pattern-action statement; the associated action is performed for each matched pattern.

An input line is made up of fields separated by white space. (This default can be changed by using FS, vide infra.) The fields are denoted $1, $2, ... ; $0 refers to the entire line.

A pattern-action statement has the form

```
pattern { action }
```

A missing { action } means print the line; a missing pattern always matches.

An action is a sequence of statements. A statement can be one of the following:

```
if ( conditional ) statement [ else statement ]
while ( conditional ) statement
for ( expression ; condition ; expression ) statement
break
continue
{[statement] ... }
variable = expression
print [ expression-list ] [ >expression ]
printf format [ , expression-list ] [ >expression ]
next      #skip remaining patterns on this input line
exit      #skip the rest of the input
```

Statements are terminated by semicolons, newlines or right braces. An empty expression-list stands for the whole line. Expressions take on string or numeric values as appropriate, and are built using the operators +, -, *, /, %, and concatenation (indicated by a blank). The C operators ++, --, +=, -=, *=, /=, and %= are also available in expressions. Variables may be scalars, array elements (denoted x[i]) or fields. Variables are initialized to the null string. Array subscripts may be any string, not necessarily numeric; this allows for a form of associative memory. String constants are quoted "...".

The print statement prints its arguments on the standard output (or on a file if >file is present), separated by the current output field separator, and terminated by the output record separator. The printf statement formats its expression list according to the format (see printf(3)).

The built-in function length returns the length of its argument taken as a string, or of the whole line if no argument. There are also built-in functions exp, log, sqrt, and int. The last truncates its argument to an integer, substr(s, m, n) returns the n-character substring of s that begins at position m. The function sprintf(fmt, expr, expr, ...) formats the expressions according to the printf(3) format given by fmt and returns the resulting string.

Patterns are arbitrary Boolean combinations (!, ||, &&, and parentheses) of regular expressions and relational expressions. Regular expressions must be surrounded by slashes and are as in egrep. Isolated regular expressions in a pattern apply to the entire line. Regular expressions may also occur in relational expressions.

A pattern may consist of two patterns separated by a comma; in this case, the action is performed for all lines between an occurrence of the first pattern and the next occurrence of the second.

A relational expression is one of the following:

```
expression matchop regular-expression
expression relop expression
```

where a relop is any of the six relational operators in C, and a matchop is either ~ (for contains) of !~ (for does not contain). A conditional is an arithmetic expression, a relational expression, or a Boolean combination of these.

The special patterns BEGIN and END may be used to capture control before the first input line is read and after the last. BEGIN must be the first pattern, END the last.

A single character c may be used to separate the fields by starting the program with

```
BEGIN { FS = c }
```

or by using the -Fc option.

Other variable names with special meanings include NF, the number of fields in the current record; NR, the ordinal number of the current record; FILENAME, the name of the current input file; OFS, the output field separator (default blank); ORS, the output record separator (default newline); and OFMT, the output format for numbers (default "%.6g").

Examples

Print lines longer than 72 characters:

```
length > 72
```

Print first two fields in opposite order:

```
{ print $2, $1 }
```

Add up first column, print sum and average:

```
        { s += $1 }
END     { print  "sum is", s, "average is", s/NR }
```

Print fields in reverse order:

```
{ for (i = NF; i > 0; --i) print $i }
```

Print all lines between start/stop pairs:

```
/start/, /stop/
```

Print all lines whose first field is different from previous one:

```
$1 != prev { print; prev = $1 }
```

See Also

grep(C), lex(CP), sed(C)
The XENIX Text Processing Guide

Notes

There are no explicit conversions between numbers and strings. To force an expression to be treated as a number add 0 to it; to force it to be treated as a string concatenate "" to it. Input white space is not preserved on output if fields are involved.

Name

basename - Delivers filename part of pathname

Syntax

basename string [suffix]

Description

Basename deletes any prefix ending '/' and the suffix, if present in string, from string, and prints the result on the standard output. It is normally used inside substitution marks `` in shell procedures.

The related command dirname delivers all but the last level of the pathname in string.

Examples

This shell procedure invoked with the argument /usr/src/cmd/cat.c compiles the named file and moves the output to a file named cat in the current directory:

```
cc $1
mv a.out `basename $1 .c`
```

See Also

dirname(C), sh(C)

Name

bsh -- Business Shell

Syntax

bsh [-fhqs] [menusystem]

Description

Bsh is a menu-driven command language interpreter. It may be installed as the "login shell" in the password file, or it may be invoked directly by the user.

The command is implemented using the termcap and curses facilities from UC Berkeley. It must be run from a terminal which is defined within /etc/termcap.

This command should only be run interactively. A user's terminal may be left in a very strange state if bsh is run in the background.

In the options described below, either "line feed" or "return" performs the newline function.

Options

-f Start bsh in "fast" mode. In this mode, a prompt whose first letter is a lower-case alphabetic or numeric character is executed immediately when the first letter is typed. The system does not wait for a terminating newline. Prompts whose first letter is upper-case alphabetic wait for a terminating newline before executing the requested actions. Fast mode is the default initial mode, if not over-ridden by the command line or the BSHINIT variable (see below). The current mode may be changed during execution through use of the "?mode" command (described below).

-h displays a short help message describing how to invoke bsh.

-q displays a one-line descriptive summary of the syntax used to invoke bsh.

-s Start bsh in "slow" mode. In this mode, all prompts must be terminated by newline before execution occurs. The current mode may be changed during execution through use of the "?mode" command (described below).

A menu system may be specified if desired. In this case, bsh utilizes the designated menu system instead of the default one (/etc/menusys.bin). Prior to use by bsh a menu system must be "digested" using the digest(C) utility. If the specified menu system does not exist or if it is not read-accessible, bsh issues an error message and terminates.

How to create a new menu system and how to update or modify an existing menu system is described in menus(5).

Commands

prompts
Typing any of the prompts on the current menu screen immediately causes the actions associated with the prompt to be executed. It is the responsibility of the menu designer to ensure that reasonable actions exist for each prompt. Selecting a prompt with no associated action causes an error message to be displayed.

An action may be any one of the following:

> Go to a specified menu
> Execute a sh(C) script
> Execute a bsh internal command (e.g., chdir(C))

menuname
Typing the name of a menu causes it to immediately become the current menu. If the menu name is misspelled, or if it does not exist in the current menu system, an error message is displayed.

newline
Typing a newline causes the immediately preceding menu to become the current one. If there is no previous menu, an error message is displayed. Bsh does not distinguish between "line feed" and "return" -- both generate a newline.

? Typing a question mark (?) causes the "help" menu associated with the current menu to be displayed. Help menus are no different from normal menus (except, perhaps, in the type of information they contain). When the current menu is named "xyz", typing a question mark is entirely equivalent to typing "xyz?"

?? Typing a pair of question marks (??) causes the bsh system help information to be displayed. It contains much the same information as is presented here.

menuname?
Typing the name of a menu followed by question mark causes the designated help menu to become the current one.

!command
The exclamation point (!) allows the user to "escape" to the standard shell (sh(C)). The command must follow the usual rules as described in the sh(C) documentation. In particular, the command may consist of a sequence of shell commands separated by semicolons -- thus several actions may be invoked. If the command is absent, sh(C) is invoked as a sub-shell with no arguments. In this case, bsh will be resumed as soon as the sub-shell terminates. (Usually, this is accomplished by sending the sub-shell an end-of-file. End-of-file is Control-d on most terminals.) You may escape to the Berkeley C shell (csh(C)) by typing "!csh."

?index
This special command causes bsh to display its internal "index" for the current menu system. The index contains the names of every accessible menu.

?mode
This special command allows the user to change from "slow" mode to "fast" mode and vice versa. The user is asked if he wishes to change to the alternate mode. If your response begins with "y" or "Y", the change is made, otherwise the current mode remains in effect.

interrupt
Bsh will immediately return to the top-level command interpreter upon receipt of an interrupt signal. Such a signal is usually generated via the DEL, RUBOUT or BREAK key.

backspace
Bsh understands the Backspace function (as obtained from /etc/termcap).

CANcel
Bsh interprets the CANcel key to mean "restart input." The CANcel key is Control-x on many of the more popular terminals.

ESCape
Typing an ESCape has the same effect as does typing CANcel.

DC2 If the screen becomes "dirty" for some reason, you can force bsh to clear it and redisplay the current contents by transmitting an ASCII "DC2." This is Control-r on most of the currently popular terminals.

q Typing a "q", "Q" or "Quit" all have the same effect: bsh is terminated. If bsh is your login shell, "quit" also results in your being logged out.

Environment

BSHINIT
The BSHINIT environment variable contains the initial value of the default mode ("fast" or "slow"). If this variable does not exist in the environment, bsh assumes "fast" mode. BSHINIT should be set by inserting the line BSHINIT="fast" or BSHINIT="slow" into your .profile file.

Note that even if bsh is designated as the "login shell" in /etc/passwd, your .profile file will be interpreted correctly. (See login(1) and sh(1).) In particular, any overriding definitions you may have for the kill and erase characters will be correctly interpreted by bsh.

Files

~/.profile	contains commands to be executed during login(1)
/etc/menusys.bin	default menu system used by bsh
/etc/passwd	used to define a user's login name, password, home directory, shell, etc.
/etc/termcap	contains terminal attribute descriptions
/usr/lib/bsh.messages	system warning and error messages

See Also

digest(C), login(C), menus(F), sh(C), termcap(F)

Diagnostics

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

Notes

Bsh probably should never allow itself to be run in the background.

Bsh should detect the fact that the current terminal is not defined in /etc/termcap and abort gracefully.

Name

cat - Concatenates and displays files.

Syntax

cat[- u] [- s] file...

Description

Cat reads each file in sequence and writes it on the standard output. If no input file is given, or if a single dash (-) is given, cat reads from the standard input. The options are:

- s Suppresses warnings about nonexistent files.

- u Causes the output to be unbuffered.

No input file may have the same name as the output file unless it is a special file.

Examples

The following example displays file on the standard output:

```
cat file
```

The following example concatenates file1 and file2 and places the result in file3:

```
cat file1 file2 >file3
```

The following example concatenates file1 and appends it to file2:

```
cat file1 >> file2
```

See Also

cp(C), pr(C)

Name

cd - Changes working directory.

Syntax

cd [directory]

Description

Cd changes the current directory to the directory specified. If no directory is specified, the value of the shell parameter $HOME is used, and the user is placed in his home directory. The argument ".." moves up from a directory to the parent directory.

The user must have search (execute) permission in all directories specified.

Examples

The following example changes the current directory to the user's home directory:

cd

The following command changes the current directory to /usr/wendy/memos/meetings:

cd /usr/wendy/memos/meetings

The following command changes the current directory from /usr/wendy/memo/meetings to /usr/wendy/accounts/overdue:

cd ../../accounts/overdue

See Also

chroot(C), chdir(S), pwd(C), sh(C)

Name

chgrp - Changes group ID.

Syntax

chgrp group file ...

Description

Chgrp changes the group ID of each file to group. The group may be either a decimal group ID or a group name found in the file /etc/group.

Files

/etc/passwd

/etc/group

See Also

chown(C), passwd(M), group(M)

Notes

Only the owner or the super-user can change the group ID of a file.

Name

chmod - Changes the access permissions of a file or directory.

Syntax

chmod mode file ...

Description

The chmod command changes the access permissions (or mode) of a specified file or directory. It is used to control file and directory access by users other than the owner and super-user. The mode may be an expression composed of letters and operators (called symbolic mode), or a number (called absolute mode).

A chmod command using symbolic mode has the form:

chmod [who] + - = [permission ...] filename

Who is one or any combination of the following letters:

a Stands for "all users". If who is not indicated on the command line, a is the default. The definition of "all users" depends on the user's umask. See umask(C).

g Stands for "group", all users who have the same group ID as the owner of the file or directory.

o Stands for "others", all users on the system.

u Stands for "user", the owner of the file or directory.

The operators are:

+ Adds permission

- Removes permission

= Assigns the indicated permission and removes all other permissions (if any) for that who. If no permission is assigned, existing permissions are removed.

Permissions can be any combination of the following letters:

x Execute (search permission for directories)

r Read

w Write

s Sets owner or group ID on execution of the file to that of the owner of the file. This permission is only useful with u or g.

t Saves text in memory upon execution. ("Sticky bit", see chmod(S)). Can only be set by the super-user.

Multiple symbolic modes may be given, separated by commas, on a single command line. See the following Examples section for sample permission settings.

A chmod command using absolute mode has the form:

chmod mode filename

where mode is an octal number constructed by performing logical OR on the the following:

4000 Set user ID on execution

2000 Set group ID on execution

1000 Sets the sticky bit (see chmod(S))

0400 Read by owner

0200 Write by owner

0100 Execute (search in directory) by owner

0040 Ready by group

0020 Write by group

0010 Execute (search in directory) by group

0004 Read by others

0002 Write by others

0001 Execute (search in directory) by others

0000 No permissions

Examples

Symbolic Mode

The following command gives all users execute permission for file:

chmod + x file

The following command removes read and write permission for group and others from file:

```
chmod go-rw file
```

The following command gives other users read and write permission for file:

```
chmod o+rw file
```

The following command gives read permission to group and other:

```
chmod g+r,o+r file
```

Absolute Mode

The following command gives all users read, write and execute permission for file:

```
chmod 0777 file
```

The following command gives read and write permission to all users for file:

```
chmod 0666 file
```

The following command gives read and write permission to the owner of file only:

```
chmod 0600 file
```

See Also

ls(C), chmod(S)

Notes

The user's umask may affect the default settings.

The user ID, group ID and sticky bit settings are only useful for binary executable files. They have no effect on shell scripts.

Name

chown - Changes owner ID

Syntax

chown owner file ...

Description

Chown changes the owner ID of the files to owner. The owner may be either a decimal user ID or a login name found in the file /etc/passwd.

Files

/etc/passwd
/etc/group

See Also

chgrp(C), chown(S), group(M), passwd(M)

Notes

Only the owner or the super user can change a file's owner or group ID.

Name

cp - Copies files.

Syntax

cp file1 file2

cp files directory

Description

There are two ways to use the cp command. With the first way, file1 is copied to file2. Under no circumstance can file1 and file2 be identical. With the second way, directory is the location of a directory into which one or more files are copied.

See Also

copy(C), cpio(C), ln(C), mv(C), rm(C), chmod(S)

Notes

Special device files can be copied. If the file is a named pipe, then the data in the pipe is copied to a regular file. Similarly, if the file is a device, then the file is read until the end-of-file is reached, and that data is copied to a regular file. It is illegal to copy a directory to a file.

Name

cron - Executes commands at specified times

Syntax

/etc/cron

Description

Cron executes commands at specified dates and times according to the instructions in the file /usr/lib/crontab. Since cron never exits, it should only be executed once. This is best done by running cron from the initialization process through the file /etc/rc.

The file crontab consists of lines of six fields each. The fields are separated by spaces or tabs. The first five are integer patterns to specify the minute (0-59), hour (0-23), day of the month (1-31), month of the year (1-12), and day of the week (1-7 with 1=monday). Each of these patterns may contain a number in the range above; two numbers separated by a minus meaning a range inclusive; a list of numbers separated by commas meaning any of the numbers; or an asterisk meaning all legal values. The sixth field is a string that is executed by the Shell at the specified times. A percent character in this field is translated to a new-line character. Only the first line (up to a % or end of line) of the command field is executed by the Shell. The other lines are made available to the command as standard input.

Crontab is examined by cron periodically to see if it has changed; if it has, cron reads it. Thus it takes only a short while for entries to become effective.

Examples

An example crontab file follows:

```
30 4 * * *  /etc/sa -s > /dev/null
0 4 * * *   calendar -
15 4 * * *  find /usr/preserve -mtime +7 -a -exec rm -f {} ;
30 4 1 1 1  /usr/lib/uucp/cleanlog
40 4 * * *  find / -name '#*' -atime +3 -exec rm -f {} ;
0,5,10,15,20,25,30,35,40,45,50,55 * * * *     /usr/lib/atrun
0,10,20,30,40,50 * * * *    /etc/dmesg - >>/usr/adm/messages
1,21,41 * * * * (echo -n ' '; date; echo ) >/dev/console
```

A history of all actions by cron can be recorded in /usr/lib/cronlog. This logging occurs only if the variable

CRONLOG in /etc/default/chron is set to YES. By default this value is set to NO and no logging occurs. If logging should be turned on, be sure to monitor the size of /usr/lib/crontab so that it doesn't unreasonably consume disk space.

Files

/usr/lib/crontab

/usr/lib/cronlog

/etc/default/cron

See Also

sh(C)

Notes

Cron reads crontab only when it has changed, but it reads the incore version of that table periodically.

Name

date - Print and set the date

Syntax

date [-cms] [yymmddhhmm [.ss]]

Description

If no argument is given, the current date and time are printed. If an argument is given, the current date is set. yy is the last two digits of the year; the first mm is the month number; dd is the day number in the month; hh is the hour number (24 hour system); the second mm is the minute number; .ss is optional and is the seconds. For example:

date 10080045

sets the date to Oct 8, 12:45 AM. The year, month, and day may be omitted, the current values being the defaults. The system operates in GMT (Greenwich Mean Time). Date takes care of the conversion to and from local standard and daylight time.

The -c option causes date to use the hardware real-time clock. Thus, date -c prints the current date and time from the hardware real-time clock, and date -c yymmddhhmm sets the real-time clock.

The -s option sets the system (i.e., the software) clock to the current time and date from the hardware real-time clock.

The -m option should be used at midnight. Its primary function is to update the year on the hardware real-time clock if it is January 1 and to make adjustments to the real-time clock if it is February 29 in a leap year. (The hardware real-time clock does not automatically increment the year on January 1, and it does not allow February 29.) If -m is specified, date waits for the hardware real-time clock to reach midnight (if it hasn't already), handles January 1 and February 29, and then sets the software system clock to the current time on the hardware real-time clock. For the -m option to work correctly, the software clock and the hardware clock should be within twelve hours of one another, and date -m should be executed approximately at midnight. Use cron(C) to execute date -m at midnight each day.

Xenix normally uses only the system (i.e., software) clock. The only time that Xenix uses the hardware real-time clock is with the date command.

Files

/usr/admwtmp to record time-setting

See Also

cron(C), utmp(M)

Diagnostics

'No permission' is you aren't the super-user and you try to change the date; 'bad conversion' if the date is syntactically incorrect; 'waiting for midnight...' if date -m is executed before the hardware clock reaches midnight.

Name

dd - Convert and copy a file

Syntax

dd [option=value] ...

Description

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

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

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

Cbs is used only if ascii or ebcdic conversion is specified. In the former case cbs characters are placed into the conversion buffer, converted to ASCII, and trailing blanks trimmed and newline added before sending the line to the

output. In the latter case ASCII characters are read into the conversion buffer, converted to EBCDIC, and blanks added to make up an output record of size cbs.

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

Examples

For example, to read an EBCDIC tape blocked ten 80-byte EBCDIC card images per record into the ASCII file x:

```
dd if=/dev/rmt0 of=x ibs=800 cbs=80 conv=ascii,lcase
```

Note the use of raw magtape. Dd is especially suited to I/O on the raw physical devices because it allows reading and writing in arbitrary record sizes.

To skip over a file before copying from magnetic tape do

```
(dd of=/dev/null; dd of=x) </dev/rmt0
```

See Also

copy(C), cp(C), tr(C)

Diagnostics

f+p records in(out): numbers of full and partial records read(written)

Notes

The ASCII/EBCDIC conversion tables are taken from the 256 character standard in the CACM Nov, 1968. The 'ibm' conversion, while less blessed as a standard, corresponds better to certain IBM print train conventions. There is no universal solution.

Newlines are inserted only on conversion to ASCII; padding is done only on conversion to EBCDIC. These should be separate options.

Name

default – Default program information directory.

Description

The files in the directory **/etc/default** contain the default information used by system commands such as *dump*(C) and *remote*(C). Default information is any information required by the command that is not explicitly given when the command is invoked.

The directory may contain zero or more files. Each file corresponds to one or more commands. A command searches a file whenever it has been invoked without sufficient information. Each file contains zero or more entries which define the default information. Each entry has the form:

keyword

or

keyword=value

where *keyword* identifies the type of information available and *value* defines its value. Both *keyword* and *value* must consist of letters, digits, and punctuation. The exact spelling of a *keyword* and the appropriate *values* depend on the command and are described with the individual commands.

Any line in a file beginning with a number sign (#) is considered a comment and is ignored.

Files

/etc/default/dump

/etc/default/dumpdir

/etc/default/lpd

/etc/default/mkuser

/etc/default/passwd

/etc/default/quot

/etc/default/micnet

/etc/default/restor

/etc/default/su

See Also

dump(C), dumpdir(C), lpr(C), mkuser(C), pwadmin(C), quot(C), remote(C), restor(C), su(C)

Name

df - Reports the number of free disk blocks.

Syntax

df[- t] [- f] [filesystem...]

Description

Df prints out the number of free blocks and free inodes available for on-line file systems by examining the counts kept in the super-blocks. One or more filesystem arguments may be specified by device name (for example, /dev/hdØ or /dev/usr). If the filesystem argument is unspecified, then the free space on all mounted file systems is sent to the standard output. The list of mounted file systems is given in /etc/mnttab.

The - t flag causes the total allocated block figures to be reported as well.

If the - f flag is given, only an actual count of the blocks in the free list is made (free inodes are not reported). With this option, df reports on raw device.

Files

/dev/*

/etc/mnttab

See Also

fsck(C), mnttab(F)

Notes

See Notes under mount(C).

Name

disable - turns off terminals.

Syntax

disable [-d] [[-e] tty ...

Description

This program manipulates the /etc/ttys file and signals init to disallow logins on a particular terminal. The -d and -e options "disable" and "enable" terminals, respectively.

Examples

A simple example follows:

```
disable tty01
```

Multiple terminals can be disabled or enabled using the -d and -e switches before the appropriate terminal name:

```
disable tty01 -e tty02 -d tty03 tty04
```

Files

/dev/tty*
/etc/ttys

See Also

login(C), enable(C), ttys(F), getty(M), init(M)

Name

du - Summarizes disk usage

Syntax

du [-ars] [names]

Description

Du gives the number of blocks contained in all files and (recursively) directories within each directory and file specified by the names argument. The block count includes the indirect blocks of the file. If names is missing, the current directory is used.

The optional argument -s causes only the grand total (for each of the specified names) to be given. The optional argument -a causes an entry to be generated for each file. Absence of either causes an entry to be generated for each directory only.

Du is normally silent about directories that cannot be read, files that cannot be opened, etc. The -r option will cause du to generate messages in such instances.

A file which has two links to it is only counted once.

Notes

If the -a option is not used, nondirectories given as arguments are not listed.

If there are too many distinct linked files, du counts the excess files more than once.

Files with holes in them will get an incorrect block count.

Name

dump - Incremental file system backup

Syntax

dump [key [arguments] filesystem]

Description

Dump copies to the specified device all files changed after a certain date in the filesystem. The key specifies the date and other options about the backup where a key consists of characters from the set 0123456789kfusd. The meanings of these characters are described below:

f Place the backup on the next argument file instead of the default device.

u If the backup completes successfully, writes the date of the beginning of the backup to the file '/etc/ddate'. This file records a separate date for each file system and each backup level.

0-9 This number is the 'backup level'. Backs up all files modified since the last date stored in the file '/etc/ddate' for the filesystem at lesser levels. If no date is determined by the level, the beginning of time is assumed; thus the option 0 causes the entire filesystem to be backed up.

s For backups to magnetic tape, the size of the tape is specified in feet. The number of feet is taken from the next argument. When the specified size is reached, dump will wait for reels to be changed. The default size is 2300 feet.

d For backups to magnetic tape, the density of the tape, expressed in BPI, is taken from the next argument. This is used in calculating the amount of tape used per write. The default is 1600.

k This option is used when backing up to a block-structured device, such as a floppy disk. The size (in K-bytes) of the volume being written is taken from the next argument. If the k argument is specified, any s and d arguments are ignored. The default is to use s and d.

If no arguments are given, the key is assumed to be 9u and a default file system is backed up to the default device.

The first backup should be a full level 0 backup:

```
dump 0u
```

Next, periodic level-9 backups should be made on an exponential progression of tapes or floppies:

```
dump 9u
```

(This is sometimes called Tower or Hanoi progression after the name of the game where a similar progression occurs, i.e., - 1 2 1 3 1 2 1 4 ... where backup 1 is used every other time, backup 2 every fourth, backup 3 every eighth, etc.) When the level-9 incremental backup becomes unmanageable because a tape is too full or too many floppies are required, a level-1 dump should be made:

```
dump 1u
```

After this, the exponential series should progress as if uninterrupted. These level-9 backups are based on the level-1 backup, which is based on the level-0 full backup. This progression of levels of backups can be carried as far as desired.

The default file system and the backup device depend on settings of the variables DISK and TAPE, respectively, in the file /etc/default/dump.

Files

/etc/ddate	Records backup dates of file system/level.
/etc/default/dump	Default dump information.

See Also

XENIX Operator's Guide

cpio(C), default(M), dump(F), dumpdir(C), restor(C)

Diagnostics

If the backup requires more than one volume, (where a volume is likely to be a floppy disk or tape), you will be asked to change volumes. Press **RETN** after changing volumes.

Notes

Sizes are based on 1600 BPI for blocked tape; the raw magnetic tape device has to be used to approach these densities. Write errors to the backup device are usually fatal. Read errors on the file system are ignored.

Warning

When backing up to floppy disks, be sure to have enough formatted floppies before you begin.

Name

dumpdir - Print the names of files on a backup archive

Syntax

dumpdir [f filename]

Description

Dumpdir is used to list the names and inode numbers of all the files and directories on archive written with the dump command. This is most useful when attempting to determine the location of a particular file in a set of backup archives.

The f option causes filename to be used as the name of the backup device instead of the default. The backup device depends on the setting of the variable TAPE in the file /etc/default/dump.

Files

Temporary files
rst*

See Also

dump(C), restor(C), default(M)

Diagnostics

If the dump extends over more than one volume, (where a volume is a floppy disk or tape) you will be asked to change volumes. Press **RETN** after changing volumes.

Name

dump.hd - dump a hard disk to tape

Syntax

/etc/dump.hd

Description

The dump.hd command dumps the entire file system from the hard disk to the cartridge tape. Dump.hd should be run in single-user mode in order to guarantee that the hard disk is not being used by any other users while dump.hd is running.

Dump.hd produces a level 0 dump tape with today's date. Refer to dump(C) for further information about dump levels.

Dump.hd only dumps the file system from the first hard disk to tape; it does not dump the second hard disk. If you want to dump the second hard disk (i.e., the add-on hard disk) to tape, use dump(C).

SEE ALSO
dump(C), dump(F), dumpdir(C), restore.hd(C), restor(C)

Name

echo - Echo arguments

Syntax

echo [-n] [-e] [--] [arg] ...

Description

Echo writes its arguments separated by blanks and terminated by a newline on the standard output.

Echo is useful for producing diagnostics in shell programs and for writing constant data on pipes. To send diagnostics to the standard error file, do 'echo ... 1>&2'.

The following options are recognized:

-n Prints line without a newline.

-e Prints arguments on the standard error output.

-- Prints arg exactly so that an argument beginning with a dash (e.g., -e or -n) can be specified.

Echo also understands C-like escape conventions. The following escape sequences need to be quoted so that the shell interprets them correctly:

\b Backspace

\c Prints line without newline; same as use of -n option

\f Form feed

\n Newline

\r Carriage return

\t Tab

\\ Backslash

\n The 8-bit character whose ASCII code is the 1-, 2- or 3-digit octal number n, which must start with a zero

Echo is useful for producing diagnostics in command files and for sending known data into a pipe.

See Also

sh(C)

Name

ed - Invokes the text editor

Syntax

ed [-] [-x] [name]

Description

Ed is the standard text editor.

If a name argument is given, ed simulates an e command (see below) on the named file; that is to say, the file is read into ed's buffer so that it can be edited. If -x is present, an x command is simulated first to handle an encrypted file. The optional - suppresses the printing of character counts by e, r, and w commands, of diagnostics from e and q commands, and of the ! prompt after a !shell command. If -x is present, an x command is simulated first to handle an encrypted file.

Ed operates on a copy of any file it is editing; changes made in the copy have no effect on the file until a w (write) command is given. The copy of the text being edited resides in a temporary file called the buffer.

Commands to ed have a simple and regular structure: zero or more addresses followed by a single character command, possibly followed by parameters to the command. These addresses specify one or more lines in the buffer. Missing addresses are supplied by default.

In general, only one command may appear on a line. Certain commands allow the addition of text to the buffer. While ed is accepting text, it is said to be in input mode. In this mode, no commands are recognized; all input is merely collected. Input mode is left by typing a period '.' alone at the beginning of a line.

Ed supports a limited form of regular expression notation. A regular expression specifies a set of strings of characters. A member of this set of strings is said to be matched by the regular expression.

The following one-character expressions match a single character:

1.1 An ordinary character (not one of those discussed in 1.2 below) is a one-character regular expression that matches itself.

1.2 A backslash (\) followed by any special character is a one-character regular expression that matches the special character itself. The special characters are:

a. ., *, [, and \ (dot, star, left square bracket, and backslash, respectively), which are always special, except when they appear within square brackets ([]; see 1.4 below).

b. ^ (caret), which is special at the beginning of an entire regular expression (see 3.1 and 3.2 below), or when it immediately follows the left of a pair of square brackets ([ ], see 1.4 below).

c. $ (dollar sign), which is special at the end of an entire regular expression (see 3.2 below).

d. The character used to bound (i.e., delimit) an entire regular expression, which is special for that regular expression (for example, see how slash (/) is used in the g command, below).

1.3 A period (.) is a one-character regular expression that matches any character except newline.

1.4 A nonempty string of characters enclosed in square brackets ([]) is a one-character regular expression that matches any one character in that string. If, however, the first character of the string is a caret (^), the one-character regular expression matches any character except newline and the remaining characters in the string. The star (*) has this special meaning only if it occurs first in the string. The dash (-) may be used to indicate a range of consecutive ASCII characters; for example, [0-9] is equivalent to [0123456789]. The dash (-) loses this special meaning if it occurs first (after an initial caret (^), if any) or last in the string. The right square bracket (]) does not terminate such a string when it is the first character within it (after an initial caret (^), if any); e.g., []a-f] matches either a right square bracket (]) or one of the letters "a" through "f" inclusive. Dot, star, left bracket, and the backslash lose their special meaning within such a string of characters.

The following rules may be used to construct regular expressions from one-character regular expressions:

2.1 A one-character regular expression matches whatever the one-character regular expression matches.

2.2 A one-character regular expression followed by a star (*) is a regular expression that matches zero or more occurrences of the one-character regular expression.

If there is any choice, the longest leftmost string that permits a match is chosen.

2.3 A one-character regular expression followed by \{m\}, \{m,\}, or \{m,n\} is a regular expression that matches a range of occurrences of the one-character regular expression. The values of m and n must be nonnegative integers less than 256; \{m\} matches exactly m occurrences; \{m,\} matches at least m occurrences; \{m,n\} matches any number of occurrences between m and n, inclusive. Whenever a choice exists, the regular expression matches as many occurrences as possible.

2.4 The concatenation of regular expressions is a regular expression that matches the concatenation of the strings matched by each component of the regular expression.

2.5 A regular expression enclosed between the character sequences \(and \) is a regular expression that matches whatever the unadorned regular expression matches. See 2.6 below for a discussion of why this is useful.

2.6 The expression \n matches the same string of characters as was matched by an expression enclosed between \(and \) earlier in the same regular expression. Here n is a digit; the subexpression specified is that beginning with the n-th occurrence of \(counting from the left. For example, the expression ^\(.*\)\1$ matches a line consisting of two repeated appearances of the same string.

Finally, an entire regular expression may be constrained to match only an initial segment or final segment of a line (or both):

3.1 A caret (^) at the beginning of an entire regular expression constrains that regular expression to match an initial segment of a line.

3.2 A dollar sign ($) at the end of an entire regular expression constrains that regular expression to match a final segment of a line. The construction ^ entire regular expression $ constrains the entire regular expression to match the entire line.

The null regular expression (e.g., //) is equivalent to the last regular expression encountered.

To understand addressing in ed it is necessary to know that any time there is a current line. Generally speaking, the current line is the last line affected by a command; how-

ever, the exact effect on the current line is discussed under the description of the command. Addresses are constructed as follows.

1. The character '.' addresses the current line.

2. The character '$' addresses the last line of the buffer.

3. A decimal number n addresses the n-th line of the buffer.

4. '`x' addresses the line marked with the name x, which must be a lower-case letter. Lines are marked with the k command described below.

5. A regular expression enclosed in slashes '/' addresses the line found by searching forward from the current line and stopping at the first line containing a string that matches the regular expression. If necessary the search wraps around to the beginning of the buffer and continues up to and including the current line, so that the entire buffer is searched.

6. A regular expression enclosed in queries '?' addresses the line found by searching backward from the current line and stopping at the first line containing a string that matches the regular expression. If necessary the search wraps around to the end of the buffer.

7. An address followed by a plus sign '+' or a minus sign '-' followed by a decimal number specifies that address plus (resp. minus) the indicated number of lines. The plus sign may be omitted.

8. If an address begins with '+' or '-' the addition or subtraction is taken with respect to the current line; e.g., '-5' is understood to mean '.-5'.

9. If an address ends with '+' or '-', then 1 is added (resp. substracted). As a consequence of this rule and rule 8, the address '-' refers to the line before the current line. Moreover, trailing '+' and '-' characters have cumulative effect, so '--' refers to the current line less 2.

10. For convenience, a comma (,) stands for the address pair 1,$, while a semicolon (;) stands for the pair .,$.

Commands may require zero, or two addresses. Commands which require no addresses regard the presence of an address as an error. Commands which accept one or two addresses assume

default addresses when insufficient are given. If more addresses are given than such a command requires, the last one or two (depending on what is accepted) are used.

Addresses are separated from each other typically by a comma ','. They may also be separated by a semicolon ';'. In this case the current line '.' is set to the previous address before the next address is interpreted. This feature can be used to determine the starting line for forward and backward searches ('/', '?'). The second address of any two-address sequence must correspond to a line following the line corresponding to the first address.

In the following list of ed commands, the default addresses are shown in parentheses. The parentheses are not part of the address, but are used to show that the given addresses are the default.

As mentioned, it is generally illegal for more than one command to appear on a line. However, most commands may be suffixed by 'p' or by 'l', in which case the current line is either printed or listed respectively in the way discussed below.

(.)a
<text>
.

The append command reads the given text and appends it after the addressed line; dot is left at the last inserted line, or, if there were no inserted lines, at the addressed line. Address 0 is legal for this command: it causes the "appended" text to be placed at the beginning of the buffer.

(.)c
<text>
.

The change command deletes the addressed lines, then accepts input text that replaces these lines; dot is left at the last line input, or, if there were none, at the first line that was not deleted.

(.,.)d

The delete command deletes the addressed lines from the buffer. The line after the last line deleted becomes the current line; if the lines deleted were originally at the end of the buffer, the new last line becomes the current line.

e file

The edit command causes the entire contents of the buffer to be deleted, and then the named file to be read in; dot is set to the last line of the buffer. If no filename is given, the currently remembered filename, if any, is used (see the f command). The number

of characters read is typed; file is remembered for possible use as a default filename in subsequent e, r, and w commands. If file begins with an exclamation (!), the rest of the line is taken to be a shell command. The output of this command is read for the e and r commands. For the w command, the file is used as the standard input for the specified command. Such a shell command is not remembered as the current filename.

E file
The Edit command is like e, except the editor does not check to see if any changes have been made to the buffer since the last w command.

f file
If file is given, the filename command changes the currently remembered filename to file; otherwise, it prints the currently remembered filename.

(1,$)g/regular-expression/command list
In the global command, the first step is to mark every line that matches the given regular expression. Then, for every such line, the given command list is executed with . initially set to that line. A single command or the first of a list of commands appears on the same line as the global command. All lines of a multiline list except the last line must be ended with a \; a, i, and c commands and associated input are permitted; the . terminating input mode may be ommitted if it would be the last line of the command list. An empty command list is equivalent to the p command. The g, G, v, and V commands are not permitted in the command list. See also Notes and the last paragraph before Files below.

(1,$)G/regular-expression/
In the interactive Global command, the first step is to mark every line that matches the given regular expression. Then, for every such line, that line is printed, dot (.) is changed to that line, and any one command (other than one of the a, c, i, g, G, v, and V commands) may be input and is executed. After the execution of that command, the next marked line is printed, and so on; a newline acts as a null command; an ampersand (&) causes the re-execution of the most recent command executed within the current invocation of G. Note that the commands input as part of the execution of the G command may address and affect any lines in the buffer. The G command can be terminated by typing an INTERRUPT.

h
The help command gives a short error message that explains the reason for the most recent ? diagnostic.

H

The Help command causes ed to enter a mode in which error messages are printed for all subsequent ? diagnostics. It will also explain the previous diagnostic if there was one. The H command alternately turns this mode on and off; it is initially on.

(.)i
<text>
.

The insert command inserts the given text before the addressed line; dot is left at the last inserted line, or if there were no inserted lines, at the addressed line. This command differs from the a command only in the placement of the input text. Address Ø is not legal for this command.

(.,.+1)j

The join command joins contiguous lines by removing the appropriate newline characters. If only one address is given, this command does nothing.

(.,.)l

The list command prints the addressed lines in an unambiguous way: a few nonprinting characters (e.g., tab, backspace) are represented by mnemonic overstrikes, all other nonprinting characters are printed in octal, and long lines are folded. An l command may be appended to any command other than e, f, r, or w.

(.,.)ma

The move command repositions the addressed line(s) after the line addressed by a. Address Ø is legal for a and causes the addressed line(s) to be moved to the beginning of the file; it is an error if address a falls within the range of moved lines; dot is left at the last line moved.

(.,.)n

The number command prints the addressed lines, preceding each line by its line number and a tab character; dot is left at the last line printed. The n command may be appended to any command other than e, f, r, or w.

(.,.)p

The print command prints the addressed lines; dot is left at the last line printed. The pr command may be appended to any command other than e, f, r, or w; for example, dp deletes the current line and prints the new current line.

P

The editor will prompt with a * for all subsequent commands. The P command alternately turns this mode on and off; it is initially on.

q

The quit command causes ed to exit. No automatic write of a file is done.

Q

The editor exits without checking if changes have been made in the buffer since the last w command.

($)r file
The read command reads in the given file after the addressed line. If no filename is given, the currently remembered filename, if any, is used (see e and f commands). The currently remembered filename is not changed unless file is the very first filename mentioned since ed was invoked. Address Ø is legal for r and causes the file to be read at the beginning of the buffer. If the read is successful, the number of characters read is typed; dot is set to the last line read in. If file begins with !, the rest of the line is taken to be a shell (sh(C)) command whose output is to be read. Such a shell command is not remembered as the current filename.

(.,.)s/regular-expression/replacement/
or
(.,.)s/regular-expression/replacement/g
The substitute command searches each addressed line for an occurrence of the specified regular expression. In each line in which a match is found, all (nonoverlapped) matched strings are replaced by the replacement if the global replacement indicator g appears after the command. If the global indicator does not appear, only the first occurrence of the matched string is replaced. It is an error for the substitution to fail on all addressed lines. Any character other than space or newline may be used instead of / to delimit the regular expression and the replacement; dot is left at the last line on which a substitution occurred.

An ampersand (&) appearing in the replacement is replaced by the string matching the regular expression on the current line. The special meaning of the ampersand in this context may be suppressed by preceding it with a backslash. The characterss \n, where n is a digit, are replaced by the text matched by the n-th regular subexpression of the specified regular expression enclosed between \(and \). When nested parenthesized subexpressions are present, n is determined by counting occurrences of \(starting from the left. When the character % is the only character in the replacement,

the replacement used in the most recent substitute command is used as the replacement in the current substitute command. The % loses its special meaning when it is in a replacement string of more than one character or is preceded by a \.

A line may be split by substituting a newline character into it. The newline in the replacement must be escaped by preceding it with a \. Such a substitution cannot be done as part of a g or v command list.

(.,.)ta
This command acts just like the m command, except that a copy of the addressed lines is placed after address a (which may be 0); dot is left at the last line of the copy.

u
The undo command nullifies the effect of the most recent command that modified anything in the buffer, namely the most recent a, c, d, g, i, j, m, r, s, t, v, G, or V command.

(1,$)v/regular-expression/command list
This command is the same as the global command g except that the command list is executed with dot initially set to every line that does not match the regular expression.

(1,$)V/regular-expression/
This command is the same as the interactive global command G except that the lines that are marked during the first step are those that do not match the regular expression.

(1,$)w file
The write command writes the addressed lines into the named file. If the file does not exist, it is created with mode 666 (readable and writeable by everyone), unless the umask setting (see sh(C)) dictates otherwise. The currently remembered filename is not changed unless file is the very first filename mentioned since ed was invoked. If no filename is given, the currently remembered filename, if any, is used (see e and f commands); dot is unchanged. If the command is successful, the number of characters written is displayed. If file begins with an exclamation (!), the rest of the line is taken to be a shell command to which the addressed lines are supplied as the standard input. Such a shell command is not remembered as the current filename.

X

A key string is demanded from the standard input. Subsequent e, r, and w commands will encrypt and decrypt the text with this key by the algorithm of crypt(C). An explicitly empty key turns off encryption.

(.)=

The line number of the addressed line is typed; dot is unchanged by this command.

!shell command

The remainder of the line after the ! is sent to the XENIX shell (sh(C)) to be interpreted as a command. Within the text of that command, the unescaped character % is replaced with the remembered filename; if a ! appears as the first character of the shell command, it is replaced with the text of the previous shell command. Thus, !! will repeat the last shell command. If any expansion is performed, the expanded line is echoed; dot it unchanged.

(.+1)

An address alone on a line causes the addressed line to be printed. A RETURN alone on a line is equivalent to .+1p. This is useful for stepping forward through the editing buffer a line at a time.

If an interrupt signal (ASCII DEL or BREAK) is sent, ed prints a question mark (?) and returns to its command level.

Some size limitations: 512 characters per line, 256 characters per global command list, 64 characters per filename,and 128K characters in the buffer. The limit on the number of lines depends on the amount of user memory.

When reading a file, ed discards ASCII NUL characters and all characters after the last newline. Files that contain characters not in the ASCII set (bit 8 on) cannot be edited by ed.

If the closing delimiter of a regular expression of a replacement string (e.g., /) would be the last character before a newline, that delimiter may be omitted, in which case the addressed line is printed. Thus the following pairs of commands are equivalent:

```
s/s1/s2s/s1/s2/p
g/s1g/s1/p
?s1?s1?
```

Files

/tmp/e#	Temporary; # is the process number
ed.hup	Work is saved here if the terminal is hung up

See Also

crypt(C), grep(C), sed(C), sh(C)

Diagnostics

?	Command errors
? file	An inaccessible file

Use the help and Help commands for detailed explanations.

If changes have been made in the buffer since the last w command that wrote the entire buffer, ed warns the user if an attempt is made to destroy ed's buffer via the e or q commands: it prints ? and allows you to continue editing. A second e or q command at this point will take effect. The dash (-) command-line option inhibits this feature.

Notes

An exclamation (!) command cannot be subject to a g or a v command.

The ! command and the ! escape from the e, r, and w commands cannot be used if the editor is invoked from a restricted shell (see sh(C)).

The sequence \n in a regular expression does not match any character.

The l command mishandles DEL.

Files encrypted directly with the crypt(C) command with the null key cannot be edited.

Because Ø is an illegal address for the w command, it is not possible to create an empty file with ed.

Name

enable - turns on terminals.

Syntax

enable [-d] [[-e] tty ...

Description

This program manipulates the /etc/ttys file and signals init to allow logins on a particular terminal. The -e and -d options may be used to allow logins on some terminals and disallow logins on other terminals in a single command.

Examples

A simple command to enable tty01 follows:

```
enable tty01
```

Multiple terminals can be disabled or enable using the -d and -e switches before the appropriate terminal name:

```
enable tty01 -e tty02 -d tty03 tty04
```

Files

/dev/tty*
/etc/ttys

See Also

login(C), disable(C), ttys(F), getty(M), init(M)

Name

environ - The user environment.

Description

The user environment is a collection of information about a user, such as his login directory, mailbox, and terminal type. The environment is stored in special "environment variables," which can be assigned character values, such as names of files, directories, and terminals. These variables are automatically made available to programs and commands invoked by the user. The commands can then use the values to access the user's files and terminal.

The following is a short list of environment variables.

PATH — Defines the search path for the directories containing commands. The system searches these directories whenever a user types a command without giving a full pathname. The search path is one or more directory names separated by colons (:). Initially, PATH is set to :/bin:/usr/bin.

HOME — Names the user's login directory. Initially, HOME is set to the login directory given in the user's **passwd** file entry.

TERM — Defines the type of terminal being used. This information is used by commands such as *more*(C) which rely on information about the capabilities of the user's terminal. The variable may be set to any valid terminal name (see *terminals*(M)) directly or by using the *tset*(C) command.

TZ — Defines time zone information. This information is used by *date*(C) to display the appropriate time. The variable may have any value of the form xxx*n*zzz where xxx is standard local time zone abbreviation, *n* is the difference in hours from GMT, and zzz is the daylight-saving local time zone abbreviation (if any). For example, EST5EDT. The difference for a location east of England can be given as a negative number.

The environment can be changed by assigning a new value to a variable. An assignment has the form

name=value

For example, the assignment

TERM=h29

sets the TERM variable to the value "h29". The new value can be "exported" to each subsequent invocation of a shell by exporting the variable with the *export* command (see *sh*(C)) or by using the *env*(C) command.

A user may also add variables to the environment, but must be sure that the new names do not conflict with exported shell variables such as MAIL, PS1, PS2, and IFS. Placing assignments in the .profile file is a useful way to change the environment automatically before a session begins.

Note that the environment is made available to all programs as a string of arrays. Each string has the form:

name=value

where the *name* is the name of an exported variable and the *value* is the variable's current value. For programs started with a *exec*(S) call, the environment is available through the external pointer *environ*. For other programs, individual variables in environment are available through *getenv*(S) calls.

See Also

env(C), login(M), sh(C), exec(S), getenv(SC), profile(M)

Name

expr - Evaluates arguments as an expression.

Syntax

expr arguments

Description

The arguments are taken as an expression. After evaluation, the result is written on the standard output. Terms of the expression must be separated by blanks. Characters special to the shell must be escaped. Note that zero is returned to indicate a zero value, rather than the null string. Strings containing blanks or other special characters should be quoted. Integer-valued arguments may be preceded by a unary minus sign. Internally, integers are treated as 32-bit, 2's complement numbers.

The operators and keywords are listed below. Expressions should be quoted by the shell, since many of the characters that have special meaning in the shell also have special meaning in expr. The list is in order of increasing precedence, with equal precedence operators grouped within braces ({ and }).

expr | expr
Returns the first expr if it is neither null nor 0, otherwise returns the second expr.

expr & expr
Returns the first expr if neither expr is null nor 0, other returns 0.

expr { =, >, >=, <, <=, != } expr
Returns the result of an integer comparison if both arguments are integers, otherwise returns the result of a lexical comparison.

expr { +, - } expr
Addition or subtraction of integer-valued arguments.

expr { *, /, % } expr
Multiplication, division, or remainder of the integer-valued arguments.

expr : expr
The matching operator : compares the first argument with the second argument which must be a regular expression; regular expression syntax is the same as that

of ed(C), except that all patterns are "anchored" (i.e., begin with a caret (^)) and therefore the caret is not a special character in that context. (Note that in the shell, the caret has the same meaning as the pipe symbol (|).) Normally the matching operator returns the number of characters matched (zero on failure). Alternatively, the \(...\) pattern symbols can be used to return a portion of the first argument.

Examples

1. a=`expr $a + 1`

   Adds 1 to the shell variable a.

2. # For $a equal to either "/usr/abc/file" or just "file"
   'expr $a : .*/\(.*\) | $a'

   Returns the last segment of a pathname (i.e., file). Watch out for the slash alone as an argument: expr will take it as the division operator (see Notes below).

3. expr $VAR : '.*'

   Returns the number of characters in $VAR.

See Also

ed(C), sh(C)

Diagnostics

As a side effect of expression evaluation, expr returns the following exit values:

0	If the expression is neither null nor zero
1	If the expression is null or zero
2	For invalid expressions

Other diagnostics include:

syntax error	For operator/operand errors
nonnumeric argument	If arithmetic is attempted on such a string

Notes

After argument processing by the shell, expr cannot tell the difference between an operator and an operand except by the

value. If $a is an equals sign (=), the command:

```
expr $a = =
```

looks like:

```
expr = = =
```

Thus the arguments are passed to expr (and will all be taken as the = operator). The following permits comparing equals signs:

```
expr X$a = X=
```

Name

false - Returns with a nonzero exit value.

Syntax

false

Description

False does nothing except return with a nonzero exit value. True(C), false's counterpart, does nothing except return with a zero exit value. False is typically used in shell procedures such as:

```
until false
do
     command
done
```

See Also

sh(C), true(C)

Diagnostics

False has exit status 1.

Name

fcopy - Copy a floppy diskette

Syntax

fcopy

Description

Fcopy is used to make duplicate copies of a floppy diskette. Fcopy is menu driven and will ask whether you wish to copy a diskette or quit. After one copy has been made, it will ask if you desire to make more copies of the same diskette. All diskettes must have been previously formatted. See format(C) to prepare diskettes (format) before making copies. Also check to verify there is enough disk space available by entering the df command.

Notes

Since the routine was written for a single floppy disk system, it copies the entire diskette to the hard disk and then copies it from the hard disk to the new diskette requiring 1440 blocks of space on the hard disk.

Files

./junk.?????? Temporary working file, created and subsequently removed by fcopy.

Name

find - Finds files.

Syntax

find pathname-list expression

Description

Find recursively descends the directory hierarchy for each pathname in the pathname-list (i.e., one or more pathnames) seeking files that match a Boolean expression written in the primaries given below. In the descriptions, the argument n is used as a decimal integer where +n means more than n, -n means less than n, and n means exactly n.

-name file — True if file matches the current filename. Normal shell argument syntax may be used if escaped (watch out for the left bracket ([), the question mark (?) and the star (*)).

-perm onum — True if the file permission flags exactly match the octal number onum (see chmod(C)). If onum is prefixed by a minus sign, more flag bits (Ø17777, see stat(S)) become significant and the flags are compared:

```
(flags&onum)==onum
```

-n — True if the file is a semaphore or shared data file.

-type c — True if the type of the file is c, where c is b, c, d, p, or f, for block special file, character special file, directory, named pipe, or plain file.

-links n — True if the file has n links.

-user uname — True if the file belongs to the user uname. If uname is numeric and does not appear as a login name in the /etc/passwd file, it is taken as a user ID.

-group gname	True if the file belongs to the group gname. If gname is numeric and does not appear in the /etc/group file, it is taken as a group ID.
-size n	True if the file is n blocks long (512 bytes per block).
-atime n	True if the file has been accessed in n days.
-mtime n	True if the file has been modified in n days.
-ctime n	True if the file has been changed in n days.
-exec cmd	True if the executed cmd returns a zero value as exit status. The end of cmd must be punctuated by an escaped semicolon. A command argument {} is replaced by the current pathname.
-ok cmd	Like -exec except that the generated command line is printed with a question mark first, and is executed only if the user responds by typing y.
-print	Always true; causes the current pathname to be printed.
-newer file	True if the current file has been modified more recently than the argument file.
(expression)	True if the parenthesized expression is true (parentheses are special to the shell and must be escaped).

The primaries may be combined using the following operators (in order of decreasing precedence):

negation
The negation of a primary is specified with the exclamation (!) unary not operator.

AND The AND operation is implied by the juxtaposition of two primaries.

OR The OR operation is specified with the -o operator given between two primaries.

Examples

The following removes all files named a.out or *.o that have not been accessed for a week:

```
find /\(-name a.out -o -name '*.o'\)-atime +7 -exec rm {} \;
```

Files

/etc/passwd

/etc/group

See Also

cpio(C), sh(C), test(C), stat(S), cpio(F)

Name

format - Format a floppy diskette while running XENIX

Syntax

format

Description

Format is a menu-driven program for formatting floppy diskettes.

For the Altos 586 computer systems, diskettes are formatted in Altos 5-1/4 inch, double-density, double-sided format.

To use the format utility, enter:

format <CR>

You are then prompted by the menu to format (and to insert a diskette) or to quit.

Name

fsck - Checks and repairs file systems.

Syntax

/etc/fsck [options] [file-system] ...

Description

Fsck audits and interactively repairs inconsistent conditions for XENIX file systems, whether XENIX version 2.3 or 3.0. If a file system is consistent then 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 concurrence before each correction is attempted. It should be noted that most corrective actions result in some loss of data. The amount and severity of the loss may be determined from the diagnostic output. The default action for each consistency correction is to wait for the operator to respond "yes" or "no". If the operator does not have write permission fsck defaults to the action of the -n option.

The following flags are interpreted by fsck:

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

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

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

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

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

If b:c is not given, then the values used when the file system was created are used. If these values were not specified, then a reasonable default value is used.

-S Conditionally reconstructs the free list. This option is like -sb:c above except that the free list is rebuilt only if there are no discrepancies discovered in

the file system. Using -S forces a "no" response to all questions asked by fsck. This option is useful for forcing free list reorganization on uncontaminated file systems.

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

-rr Recovers the root file system. The required filesystem argument must refer to the root file system, and preferably to the block device (normally /dev/root). This switch implies -y and overrides -n. If any modifications to the file system are required, the system will be automatically shutdown to insure the integrity of the file system.

-c Causes any supported file system to be converted to the type of the current file system. The user is asked to verify the request for each file system that requires conversion unless the -y option is specified. It is recommended that every file system be checked with this option, while unmounted if it is to be used with the current version of XENIX. To update the active root file system, it should be checked with:

```
fsck - c - rr/dev/root
```

If no file systems are specified, fsck reads a list of default file systems from the file /etc/checklist.

Inconsistencies checked are as follows:

- Blocks claimed by more than one inode or the free list
- Blocks claimed by an inode or the free list outside the range of the file system
- Incorrect link counts
- Size checks:

  Incorrect number of blocks
  Directory size not 16-byte aligned

- Bad inode format
- Blocks not accounted for anywhere

- Directory checks:

  File pointing to unallocated inode
  Inode number out of range

- Super-block checks:

  More than 65536 inodes
  More blocks for inodes than there are in the file system

- Bad free block list format

- Total free block or free inode count incorrect

Orphaned files and directories (allocated but unreferenced) are, with the operator's concurrence, reconnected by placing them in the lost+found directory. The name assigned is the inode number. The only restriction is that the directory lost+found must preexist in the root of the file system being checked and must have empty slots in which entries can be made. This is accomplished by making lost+found, copying a number of files to the directory, and then removing them (before fsck is executed).

Files

/etc/checklist	Contains default list of file systems to check

See Also

checklist(F), filesystem(F)

Diagnostics

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

Notes

Fsck will not run on a mounted non-raw file system unless the file system is the root file system or unless the -n option is specified and no writing out of the file system will take place. If any such attempt is made, a warning is printed and no further processing of the file system is done for the specified device.

Although checking a raw device is almost always faster, there is no way to tell if the file system is mounted. And cleaning a mounted file system will almost certainly result in an inconsistent superblock.

Warning

For XENIX 2.3 file system to be properly supported under XENIX 3.Ø, it is necessary that fsck be run on each 2.3 file system to be mounted under the XENIX 3.Ø kernel. For the root file system, "fsck - rr /dev/root" should be run and for all other file systems "fsck /dev/??" on the unmounted block device should be used.

Name

ftp - Transfer files between machines

Syntax

ftp [-f device] [-s speed] [name]

Description

Ftp allows file transfer between two Altos Computer Systems via an asynchronous serial channel. On the sending side, name is a file or list of files to be sent. If name is "-", standard input is sent. On the receiving side, name is an existing directory into which the files are received. If name is omitted, files are received into the current directory. If name is "--", received files are written to standard output.

The following options are interpreted by ftp:

-f The special file device is used to transfer files between the machines. The ports associated with the devices on each machine should be connected via a null modem cable. The default device is /dev/tty6, which uses port 6.

-s The transmission rate is set to speed. Currently supported speeds are 1200, 2400, 4800, and 9600 bits per second. The default transmission rate is 9600 baud.

Ftp is compatible with the Ftp program available for Altos CP/M and MP/M systems, so files can be transferred betwen CP/M-MP/M systems and Xenix systems. See the CP/M-MP/M documentation for details of the CP/M-MP/M Ftp.

Ftp must be run on both the sending and receiving computer. The port that ftp is running on must have login disabled (see disable(C)). Either side may be started first, but both sides must be started within about 1 minute of each other. The sending side will output 's' every few seconds until communication is established with the other side; likewise, the receiving side will output 'w' every few seconds. During file transfer, ftp will output a '*' every time a 128 byte block is successfully transmitted, and a '?' every time a block is retransmitted to overcome a transmission error.

Notes

Since MP/M and CP/M pad files with control-Z's (octal 32), control-Z's are deleted from the end of files sent to Xenix systems.

Files sent to MP/M and CP/M systems must have filenames which are legal on those systems. Files sent from MP/M and CP/M systems to Xenix systems may end up with filenames containing and sometimes ending with spaces; the Xenix shells can deal with these filenames if the entire name is enclosed in double quotes.

If the cable gets disconnected during transmission, you must wait for ftp to die (which might take up to a minute) before you can restart on the same port, otherwise the first ftp will interfere with the second.

Name

getty - Sets terminal mode.

Syntax

/etc/getty [char]

Description

Getty is invoked by *init*(M) immediately after a terminal is opened for user logins. *Getty* displays a "login:" message, then waits for the user to type a login name. While reading the name, *getty* attempts to adapt the system to the speed and type of terminal being used. Once the name is read, *getty* calls *login*(M) with the login name as the argument.

Init calls *getty* with a single character argument taken from the *ttys*(M) file entry for the terminal line. This argument determines the line speed for the terminal, and also the "login:" greeting message, which can contain control characters to initialize the terminal for proper communication.

If the user types a name and terminates it with a newline (ASCII LF) or carriage return (ASCII CR), *getty* scans the name for uppercase alphabetic characters. If only uppercase characters are found, *getty* adapts the system to map all subsequent lowercase characters into the corresponding uppercase characters. Furthermore, if the name terminates with a carriage return character, *getty* sets the terminal's serial line mode to CRMOD (see *ioctl*(S)).

If, on the other hand, the user presses the INTERRUPT key, getty writes the login message again. It also changes the serial line speed if *char* is 1 or 3 as described below. This allows the system to adapt to terminals whose line speeds vary.

After a name has been typed and scanned, *getty* passes it to *login*(M) which asks for the user's password and completes the login process.

The following arguments from the *ttys* file are understood:

- 110 baud. Intended for an ASR-33 console, for example an operator's console.

0 150 baud for an ASR-37 console.

1 Cycles through 300-150-110-1200 baud. Useful for dialup lines.

2 300-baud console decwriter.

3 Cycles through 1200-300-150-110 baud. Recommended for dialup lines.

4 2400 baud.

5 4800 baud.

6 9600 baud.

See Also

login(M), ioctl(S), ttys(M), init(M)

Name

grep, egrep, fgrep - Searches a file for a pattern.

Syntax

grep [options] expression [files]

egrep [options] [expression] [files]

fgrep [options] [strings] [files]

Description

Commands of the grep family search the input file (standard input default) for lines matching a pattern. Normally, each line found is copied to the standard output. Grep patterns are limited regular expressions in the style of ed(c); it uses a compact nondeterministic algorithm. Egrep patterns are full regular expressions; it uses a fast deterministic algorithm that sometimes needs exponential space. Fgrep patterns are fixed strings; it is fast and compact. The following options are recognized.

- v All lines but those matching are printed.

- x Prints only exact matches of an entire line. (Fgrep only.)

- c Only a count of matching lines is printed.

- l Only the names of files with matching lines are listed, separated by newlines.

- n Each line is preceded by its relative line number in the file.

- b Each line is preceded by the block number on which it was found. This is sometimes useful in locating disk block numbers by context.

- s Suppresses error messages produced for nonexistent or unreadable files.

- y Turns on matching of letters of either case in the input so that case is insignificant. Does not work for egrep.

- e expression
  Same as a simple expression argument, but useful when the expression begins with a dash (-).

- f file
  The regular expression for grep or egrep, or strings list (for fgrep) is taken from the file.

In all cases, the filename is output if there is more than one input file. Care should be taken when using the characters $, *, [, ^,], (,), and \ in expression, because they are also meaningful to the shell. It is safest to enclose the entire expression argument in single quotation marks.

Fgrep searches for lines that contain one of the strings separated by newlines.

Egrep accepts regular expressions as in ed(C), except for \(and \), with the addition of the following:

- A regular expression followed by a plus sign (+) matches one or more occurrences of the regular expression.

- A regular expression followed by a question mark (?) matches 0 or 1 occurrences of the regular expression.

- Two regular expressions separated by a vertical bar (|) or by a newline match strings that are matched by either regular expression.

- A regular expression may be enclosed in parentheses ( ) for grouping.

The order of precedence of operators is [], then *?+, then concatenation, then the backslash (\) and the newline.

See Also

ed(C), sed(C), sh(C)

Diagnostics

Exit status is 0 if any matches are found, 1 if none, 2 for syntax errors or inaccessible files.

Notes

Ideally there should be only one grep, but there isn't a single algorithm that spans a wide enough range of space-time tradeoffs.

Lines are limited to 256 characters; longer lines are truncated.

Egrep does not recognize ranges, such as [a- z], in character classes.

When using grep with the - y option, the search is not made totally case insensitive in character ranges specified within brackets.

Multiple strings can be specified in fgrep without using a separate strings file by using the quoting conventions of the shell to imbed newlines in the single string argument. For example, you might type the following at the command line:

```
fgrep string1
string2
string3'text.file
```

Similarly, multiple strings can be specified in egrep by doing:

```
egrep 'string1|string2|string3'text.file
```

Thus egrep can do almost anything that grep and frep can do.

Name

group - Format of the group file.

Description

Group contains for each group the following information:

- Group name
- Encrypted password (optional)
- Numerical group ID
- Comma-separated list of all user allowed in the group

This is an ASCII file. The fields are separated by colons; each group is separated from the next by a newline. If the password field is null, no password is demanded.

This file resides in directory /etc. Because of the encrypted passwords, it can and does have general read permission and can be used, for example, to map numerical group IDs to names.

Files

/etc/group

See Also

newgrp(C), passwd(C), crypt(S), passwd(M)

Name

haltsys - Closes out the file systems and halts the CPU.

Syntax

/etc/haltsys

Description

Haltsys does a shutdn() system call (see shutdn(S)) to flush out pending disk I/O, mark the file systems clean, and halt the processor. Haltsys takes effect immediately, so user processes should be killed beforehand. Shutdown(C) is recommended for normal system termination; it warns the users, cleans things up, and calls haltsys. Use haltsys directly only if some system problem prevents the running of shutdown.

See Also

shutdn(S), shutdown(C)

Name

init – Process control initialization.

Syntax

/etc/init

Description

The *init* program is invoked as the last step of the boot procedure and as the first step in enabling terminals for user logins. *Init* is one of three programs (*init*, *getty*(M), and *login*(M)) used to initialize a system for execution.

Init creates a process for each terminal on which a user may log in. It begins by opening the console device, /dev/console, for reading and writing. It then invokes a shell which asks for a password to start the system in maintenance mode. The user may type the password or terminate the shell by typing ASCII end-of-file (CNTRL-D) at the console. If the shell terminates, *init* performs several steps to begin normal operation. It invokes a shell and reads the commands in the /etc/rc file. This command file performs housekeeping tasks such as removing temporary files, mounting file systems, and starting daemons. Then *init* reads the file /etc/ttys and forks several times to create a process for each terminal device in the file. Each line in the /etc/ttys lists the state of the line (0 for closed, 1 for open), the line mode, and the serial line (see *ttys*(M)). Each process opens the appropriate serial line for reading and writing, assigning the file descriptors 0, 1, and 2 to the line and establishing it as the standard input, output, and error files. If the serial line is connected to a modem, the process delays opening the line until someone has dialed up and a carrier has been established on the line.

Once *init* has opened a line, it executes the *getty* program, passing the line mode as an argument. The *getty* program reads the user's name and invokes *login*(M) to complete the login process (see *getty*(M) for details). *Init* waits until the user logs out by typing ASCII end-of-file (CNTRL-D) or by hanging up. It responds by waking up and removing the former user's login entry from the file utmp, which records current users, and makes a new entry in the file wtmp, which is a history of logins and logouts. Then the corresponding line is reopened and *getty* is reinvoked.

Init has special responses to the hangup, interrupt, and quit signals. The hangup signal SIGHUP causes *init* to change the system from normal operation to maintenance mode. The interrupt signal SIGINT causes *init* to read the ttys file again to open any new lines and close lines that have been removed. The quit signal SIGQUIT causes *init* to disallow any further logins. In general, these signals have a significant effect on the system and should not be used by a naive

user. Instead, similar functions can be safely performed with the ***enable***(C), ***disable***(C), and ***shutdown***(C) commands.

Files

/dev/tty*

/etc/utmp

/usr/adm/wtmp

/etc/ttys

/etc/rc

See Also

disable(C), enable(C), login(M), kill(C), sh(C), shutdown(C), ttys(M), getty(M)

Name

kill - Terminate a process

Syntax

kill [-signo] processid ...

Description

Kill sends signal 15 (terminate) to the specified processes. If a signal number preceded by '-' is given as first argument, that signal is sent instead of terminate (see signal(S)). This will kill processes that do not catch the signal; in particular 'kill -9 ...' is a sure kill.

By convention, if process number 0 is specified, all members in the process group (i.e., processes resulting from the current login) are signaled.

The killed processes must belong to the current user unless he is the super-user.

The process number of an asynchronous process started with '&' is reported by the shell. Process numbers can also be found by using ps(C).

See Also

ps(C), sh(C)

Name

layout - Configure a hard disk

Syntax

layout layout-device cyls heads sectors swapblocks | 0

Description

Layout creates a table defining a number of "logical devices" associated with each physical disk in the XENIX system. Layout records this table on cylinder zero of each disk. Each entry in the table is in the following format:

```
struct layout {
     daddr_t l_blkoff;  /* Block offset to area */
     daddr_t l_nblocks; /* Number of blocks in area */
};
```

Layout defines ten "logical devices" on the hard disk:

0 The whole disk, with the alternate sector mechanism disabled.

1 The swap area.

2 The root file system.

3-8 Unused.

9 Alternate sector area into which bad disk sectors are automatically mapped by the XENIX kernel.

The logical device numbers correspond to device numbers in the hard disk driver.

Other device numbers are pre-defined in the XENIX kernel as follows:

10 Future expansion.

11 All of track0.

12 Boot program area.

13 Portion of cylinder zero used for fsck temporary file.

14 Layout information created by this utility.

15 Sector to sector map (see map(1)).

The cyls, heads, sectors, and swapblocks options must be specified. They represent the number of cylinders, heads, sectors per track and number of blocks in the swap area, respectively.

If swapblocks = 0, the add-on hard disk (i.e., the second hard disk on the system) is initialized.

See Also

map(C), sizefs(C)

Examples

layout /dev/hd0.layout 306 6 16 5000

For a 20 megabyte hard disk with 5000 blocks of swap area.

layout /dev/hd0.layout 512 8 16 4500

For a 40 megabyte hard disk with 4500 blocks of swap area.

layout /dev/hd1.layout 512 8 16 0

For a 40 megabyte add-on hard disk.

Name

ln - Make a link

Syntax

ln [-s] name1 [name2]

Description

A link is a directory entry referring to a file; the same file (together with its size, all its protection information, etc.) may have several links to it. There is no way to distinguish a link to a file from its original directory entry; any changes in the file are effective independently of the name by which the file is known.

Ln creates a link to an existing file name1. If name2 is given, the link has that name; otherwise it is placed in the current directory and its name is the last component of name1.

The -s option indicates that the link is a symbolic link.

See Also

rm(C)

Name

login - Gives access to the system

Syntax

login [username]

Description

The login command is used when a user initially signs on, or it may be used at any time to change from one user to another. The latter case is the one summarized above and described here. See 'How to Get Started' for how to dial up initially.

If login is invoked without an argument, it asks for a user name, and, if appropriate, a password. Echoing is turned off (if possible) during the typing of the password, so it will not appear on the written record of the session.

At some installations, an option may be invoked that will require you to enter a second "external" password. This will occur only for dial-up connections, and will be prompted by the message "External security:". Both passwords are required for a successful login.

If password aging has been invoked by the super-user on your behalf, your password may have expired. In this case, you will be shunted into passwd(C) to change it, after which you may attempt to log in again.

If you do not complete the login successfully within a certain period of time (e.g., one minute), you are likely to be returned to the "login:" prompt or silently disconnected from a dial-up line.

After a successful login, accounting files are updated, you are informed of the existence of any mail, and the start-up profile files (i.e., /etc/profile and $HOME/.profile) (if any) are executed (see profile(M)). Login initializes the user and group IDs and the working directory, then executes a command interpreter (usually sh(C)) according to specifications found in the /etc/passwd file. Argument Ø of the command interpreter is a dash (-) followed by the last component of the interpreter's pathname. The environment (see environ(M)) is initialized to:

HOME= your-login-directory

PATH=:/bin:/usr/bin

Files

/etc/utmp	accounting
/usr/adm/wtmp	accounting
/usr/mail/*	mail
/etc/motd	message-of-the-day
/etc/passwd	password file

See Also

newgrp(C), sh(C), getty(M), mail(C), passwd(C), passwd(M), profile(M), environ(M)

Diagnostics

'Login incorrect,' if the name of the password is bad.
'No Shell', 'cannot open password file', 'no directory': consult a programming counselor.
'Your password has expired. Choose a new one': password aging is implemented and yours has expired.

Name

lpd - Line printer daemon

Syntax

lpd [N]

Description

Lpd is the line printer daemon which supports multiple printer spooling. Lpd is executed automatically by the lpr(C) command. A single daemon is used per printer device, and daemons are invoked only if there is currently no daemon active. Lpd does not engage in any filtering of the data to the printer, hence, printer control codes, escape sequences and other binary will be faithfully reproduced. For serial printers, lpr supplies lpd with a tty modes setting which is non-destructively used to print individual requests. Lpd restores tty modes between each request, and at exit time.

"N" is a single numeric digit which selects a spool directory and printer device. If N were specified as "2", /usr/spool/lpd2 and /dev/lp2 would be selected. If no printer digit is supplied, Lpd assumes /dev/lp and /usr/spool/lpd. lpr(C) invokes lpd with an appropriate printer selector digit.

Lpr decides whether to invoke the lpd daemon based on the presence (or absence) of a "lock" file in each spool directory. A daemon will run until there is no more output for its printer. It also removes its lock file so that a new daemon may be started up. If the daemon were to terminate before removing its lock file, the lock file must be removed from its spool directory before printing can be resumed. Lpd prints an optional header (specified in lpr (C)), followed by a sequence of files (each followed by a formfeed).

Files

/usr/spool/lpd?	Spool directories
/dev/lp*	printer devices
/usr/spool/lpd?/lock	lock file

See Also

lpr(C), printers(M)

Name

lpr - Multiple device print spooler

Syntax

```
lpr[K]

[-sCONF]
[-SMODES]
[-n[NETNAME]] | [-@[NETNAME]]
[-m[LOGIN]]
[-pK]
[-b[X]]
[-N]
[-r]
     [file...]
```

Description

Lpr enables the user to do background printing on one of several line printers. If no [file...] is given, standard input is used.

This version of **lpr** uses a printer configuration file to support spooling through WorkNet and tty mode setting by the print daemon (see lpd(M)). See printers(M) for further information on the format of the printer configuration file. Most of the options described below have an associated environment variable and a corresponding field in a configuration file line. Command-line arguments have precedence over environment variables, and environment variables override configuration file fields. The pathname of the configuration file may be supplied to **Lpr** through the environment variable PCFILE.

Lpr miscellaneous options are:

-m[LOGNAME] -- mail user.

Send mail to a user telling when a line printer job has completed. If the option is immediately followed by a login name, this user is notified, otherwise, the user submitting the request is notified.

-bX -- Tell **lpr** to use banner

This option prepends a banner page to print request. The X argument (4 characters maximum) supplies the text of the header. Use of this option overrides the BANNER environment variable.

Use of the BANNER environment variable tells **lpd(C)** and **lpr** that BANNER is the name used in the header page attached to your job. It can be overridden using the -bX option mentioned above. If BANNER is not defined and the -b option is used, but X is not specified, the user's login name is used. A banner page will be prepended to a print request only if the -b option is specified. If it is, a page will be ejected before the banner page is printed, and no formfeed will be supplied after the print job is finished. If no banner is desired, printing will begin at the current position of the printhead (no initial formfeed), and a page will be ejected after printing is completed.

Lpr configuration options are:

-sCONF -- select a printer configuration.

This option is used to select a printer configuration line from the printer configuration file (see printers(M)). CONF is a alphanumeric string which is used to pick out a single configuration line from the configuration file. The environment variable PCONF is also used to supply this tag.

-SMODES -- daemon sets tty modes.

This option is used to supply baud and other tty modes for serial printers. The MODES argument consists of a set of tty modes, enclosed in quotes. Use the same format as for stty(C). This option overrides a mode selection from the configuration file. The environment variable PMODES may also be used in this context.

-n [NETNAME] or -@[NETNAME] -- spool through WorkNet.

The NETNAME argument is a WorkNet machine name. The environment variable PNET is also used to specify a machine name. When you use this option, files are copied to spool directories on the named machine. A print daemon is remotely invoked to do the actual printing. If no machine name is specified, printing is assumed to be local.

-pK or lpr[K] -- print on K,

where K is a single digit number which represents the printer device. If this option is not given, the environment variable PRINTER is used. If PRINTER is not defined, the configuration file is consulted.

The printer digit, K, is a single numeric character which selects a spool directory and a printer device. The default printer spool directory is /usr/spool/lpd and the default printer device is /dev/lp. If printer 2 is selected, the spool directory is /usr/spool/lpd2, and device /dev/lp2.

The printer digit may be supplied as part of the lpr name, or by the -p switch. The -p switch overrides a printer digit supplied from the file name.

-N -- suppresses the formfeed after each file.

-r -- Removes a file after it's been sent to the spool queue.

Printer Configuration:

The printer configuration file is used by lpr and other programs to provide access to printers on WorkNet, and to set the selected tty modes. The configuration file contains at least one line for each printer, in this format:

lp[N]: LINENAME: PRINTER-TYPE: [MACHINE-NAME]:[TTY MODES]

lp[N] is the device name of the printer. When you list printers on other networked machines in your configuration file, be sure you use the device name by which a printer is known on its home machine.

LINE-NAME is an alphanumeric string. Each line in your file must have a unique line name. This is the name you append to the lpr command to select a particular printer and group of tty settings.

PRINTER-NAME is set and used by the word processor Alwrite. If you do not have Alwrite, fill this field with any string.

MACHINE-NAME is the name of the machine on the network that is connected to the requested printer. If no machine name is given, printing occurs locally.

TTY MODES are baud rate and other settings selected for this print request.

If device name, tty modes, and (possibly null) machine name are specified, either by environment variable or command-line, the print spooler does not open the configuration file, since it has no need of any information contained in it.

Configuration Selection:

The following paragraphs describe how lpr selects information from the configuration file and how various options are used. Specification of an option may come from the command-line or environment variables.

When a configuration line name is specified, the print spooler goes to that line for the other information it needs. We recommend calling lpr with a line name.

If a configuration line name is specified and a printer device is also specified, the print spooler will use the first line in the printer configuration file which matches both printer device and name fields.

If a printer device alone is specified, the first line that matches the printer device is used. If the spooler is invoked as "lpr" or if no configuration file exists, the printer /dev/lp is the default.

Files

/usr/spool/lpd*	spooling directories
/dev/lp*	line printer devices
/usr/lib/lpd	line printer daemon
/bin/mail	system mailer
/etc/printers	printer configuration file

See Also

mail(C), lpd(M), printers(M)

Name

ls - List contents of directory

Syntax

ls [-ltasdrucifgmnlCqbxFRAL] [Filenames]

Description

For each directory argument, ls lists the contents of the directory; for each file argument, ls repeats its name and any other information requested. The output is sorted alphabetically by default. When no argument is given, the current directory is listed. When several arguments are given, the arguments are first sorted appropriately, but file arguments appear before directories and their contents.

There are three major listing formats. The format chosen depends on whether the output is going to a teletype, and may also be controlled by option flags. The default format for a teletype is to list the contents of directories in multi-column format, with the entries sorted down the columns. (Files which are not the contents of a directory being interpreted are always sorted across the page rather than down the page in columns. This is because the individual file names may be arbitrarily long.) If the standard output is not a teletype, the default format is to list one entry per line. Finally, there is a stream output format in which files are listed across the page, separated by "," characters. The -m flag enables this format; when invoked as l this format is also used.

The following options are available:

-l List in long format, giving mode, number of links, owner, size in bytes, and time of last modification for each file. (See below.) If the file is a special file, the size field contain instead the major and minor device numbers.

-t Sort by time modified (latest first) instead of by name, as is normal.

-a List all entries; usually '.' and '..' are not suppressed.

-s Give size in blocks, including indirect blocks, for each entry and total blocks.

-d If argument is a directory, list only its name, not its contents (mostly used with -l to get status on directory).

-r Reverse the order of sort to get reverse alphabetic or oldest first as appropriate.

-u Use time of last access instead of last modification for sorting (-t) or printing (-l).

-c Use time of file creation for sorting or printing.

-i Print i-number in first column of the report for each file listed.

-f Force each argument to be interpreted as a directory and list the name found in each slot. This option turns off -l, -t, -s, and -r, and turns on -a; the order is the order in which entries appear in the directory.

-g Give group ID instead of owner ID in long listing.

-m Force stream output format.

-n List in long format (similary to l option), except that it lists user number rather than file owner.

-l Force one entry per line output format, e.g., to a teletype.

-C Force multi-column output, e.g., to a file or a pipe.

-q Force printing of non-graphic characters in file names as the character '?'; this normally happens only if the output device is a teletype.

-b Force printing of non-graphic characters to be in the `ddd notation in octal.

-x Force columnar printing to be sorted across rather than down the page; this is the default if the last character of the name the program is invoked with is an 'x'.

-F Cause directories to be marked with a trailing '/' and executable files to be marked with a trailing '*'; this is the default if the last character of the name the program is invoked with is a 'f'.

-R Recursively list subdirectories encountered.

-A List all entries; usually '.' and '..' are suppressed.

-L Cause directories or files to be marked with a trailing '>' if they are symbolic links.

The mode printed under the -l option contains 11 characters which are interpreted as follows: the first character is

d if the entry is a directory;
b if the entry is a block-type special file;
c if the entry is a character-type special file;
m if the entry is a multiplexor-type character special file;
- if the entry is a plain file.

The next nine characters are interpreted as three sets of three bits each. The first set refers to owner permissions; the next to permissions to others in the same user-group; and the last to all others. Within each set the three characters indicate permission respectively to read, to write, or to execute the file as a program. For a directory, "execute" permission is interpreted to mean permission to search the directory for a specified file. The permissions are indicated as follows:

r if the file is readable;
w if the file is writable;
x if the file is executable;
- if the indicated permission is not granted.

The group-execute permission character is given as s if the file has set-group-ID mode; likewise, the user-execute permission character is given as s if the file has set-user-ID mode.

The last character of the mode (normally 'x' or '-') is "t" if the 1000 bit of the mode is on. See chmod(1) for the meaning of this mode and instructions on changing the file mode.

When the sizes of the files in a directory are listed, a total count of blocks, including indirect blocks is printed.

Files

/etc/passwd to get user ID's for 'ls -l'
/etc/group to get group ID's for 'ls -g'

Notes

Newline and tab are considered printing characters in file names.

The output device is assumed to be 80 columns wide.

The option setting based on whether the output is a teletype is undesirable as "ls -s" is much different than "ls -s|lpr". On the other hand, not using this setting would make old shell scripts which used ls ineffective.

Name

mail - Send or receive mail among users

Syntax

```
mail person ...
mail [ -r ] [ -q ] [ -p ] [ -f file ]
```

Description

Mail with no argument prints a user's mail, message-by-message, in last-in, first-out order; the optional argument -r causes first-in, first-out order. If the -p flag is given, the mail is printed with no questions asked; otherwise, for each message, mail reads a line from the standard input to direct disposition of the message.

newline
: Go on to next message.

d
: Delete message and go on to the next.

p
: Print message again.

-
: Go back to previous message.

s [file] ...
: Save the message in the named files ('mbox' default).

w [file] ...
: Save the message, without a header, in the named files ('mbox' default).

m [person] ...
: Mail the message to the named persons (yourself is default).

EOT (control-D)
: Put unexamined mail back in the mailbox and stop.

q
: Same as EOT.

x
: Exit, without changing the mailbox file.

!command
: Escape to the Shell to do command.

?
: Print a command summary.

An interrupt stops the printing of the current letter. The optional argument -q causes mail to exit after interrupts without changing the mailbox.

When persons are named, mail takes the standard input up to an end-of-file (or a line with just '.') and adds it to each person's 'mail' file. The message is preceded by the sender's name and a postmakr. Lines that look like postmarks are prepended with '>'. A person is usually a user name recognized by login(1). To denote a recipient on a remote system, prefix person by the system name and exclamation mark (see uucp(1)).

The -f option causes the named file, e.g., 'mbox', to be printed as if it were the mail file.

Each user owns his own mailbox, which is by default generally readable but not writable. The command does not delete an empty mailbox nor change its mode, so a user may make it unreadable if desired.

When a user logs in he is informed of the presence of mail.

Files

/usr/spool/mail/*	mailboxes
/etc/passwd	to identify sender and locate persons
mbox	saved mail
/tmp/ma*	temp file
dead.letter	unmailable text

See Also

xsend(C), write(C), uucp(C), uux(C)

Notes

There is a locking mechanism intended to prevent two senders from accessing the same mailbox, but it is not perfect and races are possible.

Name

map - Create an alternate sector map for a hard disk drive

Syntax

map layout mapfile drive

Description

Map creates a bad sector map, on mapfile, using the layout information, in layout, created by layout(1). The last argument is the logical device name which references the whole drive.

The standard invocation is:

```
map /dev/hdØ.layout  /dev/hdØ.secmap  /dev/hdØ
```

The structure used for the bad sector to alternate sector mapping is as follows:

```
struct mapsec {
     int   bad_cyl;  /* Cylinder number of bad sector */
     char  bad_hed;  /* Head number of bad sector */
     char  bad_sec;  /* Sector number of bad sector */
     int   bad_good; /* Offset into alternate sector
                        area */
};
```

This structure provides a way for the XENIX hard disk driver to recover from bad sectors it encounters when reading the hard disk. If a bad sector is read, a search of a table of the above structures is made. If an exact match of cylinder, head and sector is found, the corresponding offset is used as an index into the area reserved on the disk for alternate sectors.

See Also

layout(C), sizefs(C)

Name

mem, kmem – Memory image file.

Description

The **mem** file provides access to the computer's physical memory. All byte addresses in the file are interpreted as memory addresses. Thus, memory locations can be examined in the same way as individual bytes in a file. Note that accessing a nonexistent location causes an error.

The **kmem** file is the same as **mem** except that it corresponds to kernel virtual memory rather than physical memory.

In rare cases, the **mem** and **kmem** files may be used to write to memory and memory-mapped devices. Such patching is not intended for the naive user and may lead to a system crash if not conducted properly. Patching device registers is likely to lead to unexpected results if the device has read-only or write-only bits.

Files

/dev/mem

/dev/kmem

Name

messages – Description of system console messages.

Description

This section describes the various system messages which may appear on the system console. The messages are categorized as follows:

Fatal
Recovery is impossible.

System inconsistency
A contradictory situation exists in the kernel.

Abnormal
A probably legitimate but extreme situation exists.

Hardware
Indicates a hardware problem.

Fatal system messages begin with "panic:" and indicate hardware problems or kernel inconsistencies that are too severe for continued operation. After displaying a fatal message, the system will stop. Rebooting is required.

System inconsistency messages indicate problems usually traceable to hardware malfunction, such as memory failure. These messages rarely occur since associated hardware problems are generally detected before such an inconsistency can occur.

Abnormal messages represent kernel operation problems, such as the overflow of critical tables. It takes extreme situations to bring these problems about, so they should never occur in normal system use.

Hardware messages normally specify the device, *dev*, that caused the error. Each message gives a device specification of the form *nn/mm* where *nn* is the major number of the device, and *mm* is its minor number. The command pipeline

ls -l /dev |grep *nn* |grep *mm*

may be used to list the name of the device associated with the given major and minor numbers.

System Messages

** ABNORMAL System Shutdown **
This message appears when errors occur during normal system shutdown. It is usually accompanied by other system messages. *System inconsistency, fatal*

bad block on dev *nn/mm*
A nonexistent disk block was found on, or is being inserted in, the structure's free list. *System inconsistency.*

bad count on dev *nn/mm*
A structural inconsistency in the superblock of a file system. The system attempts a repair, but this message will probably be followed by more complaints about this file system. *System inconsistency.*

Bad free count on dev *nn/mm*
A structural inconsistency in the superblock of a file system. The system attempts a repair, but this message will probably be followed by more complaints about this file system. *System inconsistency.*

error on dev *name* (*nn/mm*)
This is the way that most device driver diagnostic messages start. The message will indicate the specific driver and complaint. The *name* is a word identifying the device.

iaddress > 2^24
This indicates an attempted reference to an illegal block number, one so large that it could only occur on a file system larger than 8 billion bytes. *Abnormal.*

Inode table overflow
Each open file requires an inode entry to be kept in memory. When this table overflows the specific request (usually *open*(S) or *creat*(S)) is refused. Although not fatal to the system, this event may damage the operation of various spoolers, daemons, the mailer, and other important utilities. Anomalous results and missing data files are a common result. *Abnormal.*

interrupt from unknown device, vec=*xxxx*
The CPU received an interrupt via a supposedly unused vector. This message is followed by "panic: unknown interrupt." Typically this event comes about when a hardware failure miscomputes the vector of a valid interrupt. *Hardware.*

no file
There are too many open files, the system has run out of entries in its "open file" table. The warnings given for the message "inode table overflow" apply here. *Abnormal.*

no space on dev *nn/mm*
This message means that the specified file system has run out of free blocks. Although not normally as serious, the warnings discussed for "inode table overflow" apply: often programs are written casually and ignore the error code returned when they tried to write to the disk; this results in missing data and "holes" in data files. The system administrator should keep close watch on the amount of free disk space and take steps to avoid this situation. *Abnormal.*

** Normal System Shutdown **
This message appears when the system has been shutdown properly. It indicates that the machine may now be rebooted or powered down.

Out of inodes on dev *nn/mm*
The indicated file system has run out of free inodes. The number of inodes available on a file system is determined when *mkfs*(C) is run. The default number is quite generous, this message should be very rare. The only recourse is to remove some worthless files from that file system, or dump the entire system to a backup device, rerun *mkfs*(C) with more inodes specified, and restore the files from backup. *Abnormal.*

out of text
When programs linked with the *ld* – i or – n switch are run, a table entry is made so that only one copy of the pure text will be in memory even if there are multiple copies of the program running. This message appears when this table is full. The system refuses to run the program which caused the overflow. Note that there is only one entry in this table for each different pure text program. Multiple copies of one program will not require multiple table entries. Each "sticky" program (see *chmod*(C)) requires a permanent entry in this table; nonsticky pure text programs require an entry only when there is at least one copy being executed. *Abnormal.*

panic: bad 287 int
Attempted execution of a real mode 287 instruction. *System inconsistency, fatal.*

panic: blkdev
An internal disk I/O request, already verified as valid, is discovered to be refering to a nonexistent disk. *System inconsistency, fatal.*

panic: devtab
An internal disk I/O request, already verified as valid, is discovered to be refering to a nonexistent disk. *System inconsistency, fatal.*

panic: iinit
The super-block of the root file system could not be read. This

message occurs only at boot time. *Hardware, fatal.*

panic: IO err in swap
A fatal I/O error occurred while reading or writing the swap area. *Hardware, fatal.*

panic: memory failure - parity error
A hardware memory failure trap has been taken. *System inconsistency, fatal.*

panic: memory management failure
An error occurred during memory management operations. *System inconsistency, fatal.*

panic: no fs
A file system descriptor has disappeared from its table. *System inconsistency, fatal.*

panic: no imt
A mounted file system has disappeared from the mount table. *System inconsistency, fatal.*

panic: no procs
Each user is limited in the amount of simultaneous processes he can have; an attempt to create a new process when none is available or when the user's limit is exceeded is refused. That is an occasional event and produces no console messages; this panic occurs when the kernel has certified that a free process table entry is available and yet can't find one when it goes to get it. *System inconsistency, fatal.*

panic: Out of swap
There is insufficient space on the swap disk to hold a task. The system refuses to create tasks when it feels there is insufficient disk space, but it is possible to create situations to fool this mechanism. *Abnormal, fatal.*

panic: general protection trap
General protection trap taken in kernel. *System inconsistency, fatal.*

panic: segment not present
An attempt has been made to access an invalid segment. It may also indicate the segment-not-present trap has been taken in the kernel. *System inconsistency, fatal.*

panic: Timeout table overflow
The timeout table is full. Timeout requests are generated by device drivers, there should usually be room for one entry per system serial line plus ten more for other usages.

panic: Trap in system
The CPU has generated an illegal instruction trap while

executing kernel or device driver code. This message is preceded with an information dump describing the trap. *System inconsistency, fatal.*

panic: Invalid TSS
Internal tables have become corrupted. *System inconsistency, fatal.*

panic: unknown interrupt
The CPU received an interrupt via a supposedly unused vector. Typically this event comes about when a hardware failure miscomputes the vector of a valid interrupt. *Hardware, fatal.*

proc on q
The system attempts to queue a process already on the process ready-to-run queue. *System inconsistency, fatal.*

Trap *type*
This message precedes a "panic:" message. The *type* is the trap number given by the processor. The message is followed by a dump of registers. *System inconsistency, fatal.*

Name

mkdir - Make a directory

Syntax

mkdir dirname ...

Description

Mkdir creates specified directories in mode 777. Standard entries, '.', for the directory itself, and '..' for its parent, are made automatically.

Mkdir requires write permission in the parent directory. The permissions assigned to the new directory are modified by the current file creation mask set by umask(C).

See Also

rm(C), umask(C)

Diagnostics

Mkdir returns exit code Ø if all directories were successfully made. Otherwise it prints a diagnostic and returns nonzero.

Name

mkfs - Construct a file system

Syntax

/etc/mkfs special proto

Description

Mkfs constructs a file system by writing on the special file special 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 block 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 i-nodes 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(C).

Two decimal number tokens come after the mode; they specify the user and group ID's 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, mkfs 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 $.

If the prototype file cannot be opened and its name consists of a string of digits, mkfs builds a file system with a single empty directory on it. The size of the file system is the value of proto interpreted as a decimal number. The number of i-nodes is calculated as a function of the filesystem size. The boot program is left uninitialized.

Mkfs can also be used to create a file system image in a regular file, rather than on a special device file, by giving the pathname of the target file, instead of special.

If the target file is not a regular file, then mkfs checks for an existing file system on that device. If it appears the device contains a file system, operator confirmation is requested before overwriting the data. The -y "yes" switch overrides this, and writes over any existing data without question. The -n switch causes mkfs to terminate without question if the target contains an existing file system. The check used is to read block one from the target device (block one is the super block) and see whether the bytes are all the same. If they are not, this is taken to be meaningful data, and confirmation is requested.

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
     $
$
```

See Also

filsys(F), dir(F)

Notes

There should be some way to specify links.

Name

mknod - Build special file

Syntax

/etc/mknod name [c] [b] major minor

/etc/mknod name p

/etc/mknod name s

/etc/mknod name m

Description

Mknod makes a special file. The first argument is the name of the entry. The second is b if the special file is block-type (disks, tape) or c if it is character-type (other devices). The last two arguments are numbers specifying the major device type and the minor device (e.g., unit, drive, or line number).

The assignment of major device numbers is specific to each system. They have to be dug out of the system source file conf.c.

Mknod can also be used to create named pipes with the p option; semaphores with the s option; and shared data (memory) with the m option.

See Also

mknod(S)

Name

modem - Set up tty port to be used with a modem

unmodem - Unset modem port

Syntax

/etc/modem /dev/ttyn
/etc/unmodem /dev/ttyn

Description

The modem command is used to set up /dev/ttyn to be used with a compatible modem. The modem command should be executed for every port that has a modem attached, every time the system is booted.

The modem command ensures that a dial-up tty will be logged out if the user simply hangs up while logged onto the system.

The XENIX tty device ignores the state of the RTS signal (Pin 4) by default. This works well with terminals, which may become momentarily disconnected. However, modems must be made aware of the RTS signal to maintain system security. The use of the modem command ensures that a particular tty will be logged out if the RTS signal goes inactive.

The unmodem command does the opposite of the modem command, i.e., it tells the kernel to ignore the state of the RTS signal.

See Also

disable(C), tty(M)

Examples

/etc/modem /dev/tty3
/etc/modem tty3

These commands are equivalent and tell the system that a modem is being used on serial port 3.

Name

more - Views a file one screen full at a time.

Syntax

more [-cdflsur] [-n] [+linenumber] [+/pattern] [name ...]

Description

This filter allows examination of a continuous text one screen full at a time. It normally pauses after each screen full, printing "--More--" at the bottom of the screen. If the user then types a carriage return, one more line is displayed. If the user hits the SPACE bar, another screen full is displayed. Other possibilities are described below.

The command line options are:

-n An integer which is the size (in lines) of the window which more will use instead of the default.

-c More draws each page by beginning at the top of the screen and erasing each line just before it draws on it. This avoids scrolling the screen, making it easier to read while more is writing. This option is ignored if the terminal does not have the ability to clear to the end of a line.

-d More prompts with the message "Hit space to continue, Rubout to abort" at the end of each screen full. This is useful if more is being used as a filter in some setting, such as a class, where many users may be unsophisticated.

-f This option causes more to count logical, rather than screen lines. That is, long lines are not folded. This options is recommended if nroff output is being piped through ul, since the latter may generate escape sequences. These escape sequences contain characters that would ordinarily occupy screen positions, but that do not print when they are sent to the terminal as part of an escape sequence. Thus more may think that lines are longer than they actually are and fold lines erroneously.

-l Does not treat CNTRL-L (form feed) specially. If this option is not given, more pauses after any line that contains a CNTRL-L, as if the end of a screen full had been reached. Also, if a file begins with a form feed, the screen is cleared before the file is printed.

-s Squeezes multiple blank lines from the output, producing only one blank line. Especially helpful when viewing nroff output, this option maximizes the useful information present on the screen.

-u Normally, more handles underlining, such as that produced by nroff in a manner appropriate to the particular terminal: if the terminal can perform underlining or has a stand-out mode, more outputs appropriate escape sequences to enable underlining or stand-out mode for underlined information in the source file. The -u option suppresses this processing.

-r Normally, more ignores control characters that it does not interpret in some way. The -r option causes these to be displayed as ^C where "C" stands for any such character.

-w Normally, more exits when it comes to the end of its input. With -w however, more prompts and waits for any key to be struck before exiting.

+linenumber
Start up at linenumber.

+/pattern
Starts up two lines before the line containing the regular expression pattern.

More looks in the file /etc/termcap to determine terminal characteristics, and to determine the default window size. On a terminal capable of displaying 24 lines, the default window size is 22 lines.

More looks in the environment variable MORE to preset any flags desired. For example, if you prefer to view files using the -c mode of operation, the shell command "MORE=-c" in the .profile file causes all invocations of more to use this mode.

If more is reading from a file, rather than a pipe, then a percentage is displayed along with the "--More--" prompt. This gives the fraction of the file (in characters, not lines) that has been read so far.

Other sequences which may be typed when more pauses, and their effects, are as follows (i is an optional integer argument, defaulting to 1):

i<space>
Displays i more lines, (or another screen full if no argument is given).

CNTRL-D
Displays 11 more lines (a "scroll"). If i is given, then the scroll size is set to i.

d Same as CNTRL-D.

iz Same as typing a space except that i, if present, becomes the new window size.

is Skips i lines and prints a screen full of lines.

if Skips i screen fulls and prints a screen full of lines.

q or Q
Exits from more.

= Displays the current line number.

v Starts up the screen editor vi at the current line. (Note that vi may not be available with your system.)

h or ?
Help command; Gives a description of all the more commands.

i/expr
Searches for the ith occurrence of the regular expression expr. If there are less than i occurrences of expr, and the input is a file (rather than a pipe), then the position in the file remains unchanged. Otherwise, a screen full is displayed, starting two lines before the place where the expression was found. The user's erase and kill characters may be used to edit the regular expression. Erasing back past the first column cancels the search command.

in Searches for the ith occurrence of the last regular expression entered.

' (Single quotation mark) Goes to the point from which the last search started. If no search has been performed in the current file, this command goes back to the beginning of the file.

!command
Invokes a shell with command. The characters % and ! in "command" are replaced with the current filename and the previous shell command respectively. If there is no current file name, % is not expanded. The sequences "\%" and "\!" are replaced by "%" and "!" respectively.

i:n Skips to the ith next file given in the command line (skips to last file if n doesn't make sense).

i:p Skips to the ith previous file given in the command line. If this command is given in the middle of printing out a file, more goes back to the beginning of the file. If i doesn't make sense, more skips back to the first file. If more is not reading from a file, the bell rings and nothing else happens.

:f Displays the current filename and line number.

:q or :Q
Exits from more (same as q or Q).

. Repeats the previous command.

The commands take effect immediately, i.e., it is not necessary to type a carriage return. Up to the time when the command character itself is given, the user may hit the line kill character to cancel the numerical argument being formed. In addition, the user may hit the erase character to redisplay the "--More--(xx%)" message.

The terminal is set to noecho mode by this program so that the output can be continuous. What you type will thus not show on your terminal, except for the slash (/) and exclamation (!) commands.

If the standard output is not a teletype, more acts just like cat, except that a header is printed before each file (if there is more than one).

A sample usage of more in previewing nroff output would be

```
nroff -ms +2 doc.n|more -s
```

Files

/etc/termcap	Terminal data base
/usr/lib/more.help	Help file

See Also

csh(C), sh(C), environ(M)

Notes

The vi and help options may not be available.

Before displaying a file, more attempts to detect whether it is a nonprintable binary file such as a directory or

executing binary image. If more concludes that a file is unprintable, it rightly refuses to print it. However, more cannot detect all possible kinds of non-printable files.

Name

mount, umount - Mount and dismount file system

Syntax

/etc/mount [special name [-r]]

/etc/umount special

Description

Mount announces to the system that a removable file system is present on special device. The file structure is mounted on directory. The directory must already exist; it becomes the name of the root of the newly mounted file structure.

The mount and umount commands maintain a table of mounted devices. If invoked with no arguments, for each special device mount prints the name of the device, the directory name of the mounted file structure, whether the file structure is readonly, and the date it was mounted.

The optional last argument indicates that the file is to be mounted read-only. Physically write-protected must be mounted in this way or errors occur when access times are updated, whether or not any explicit write is attempted.

Umount removes the removable file structure previously mounted on device special-device.

Files

/etc/mnttab: mount table

See Also

mount(S), mnttab(F)

Diagnostics

Mount issues a warning if the file structure to be mounted is currently mounted under another name.

Busy file structure cannot be dismounted with umount. A file structure is busy if it contains an open file or some user's working directory.

Notes

Some degree of validation is done on the file structure, however it is generally unwise to mount corrupt file structures.

Be warned that when in single-user mode, the commands that look in /etc/mnttab for default arguments (for example df, ncheck, quot, mount, and umount) given either incorrect results (due to a corrupt /etc/mnttab from a non-shutdown stoppage) or no results (due to an empty mnttab from a shutdown stoppage).

When multiuser this is not a problem; /etc/rc initializes /etc/mnttab to contain only /dev/root and subsequent mounts update is appropriately.

The mount(C) and umount(C) commands use a lock file to guarantee exclusive access to /etc/mnttab, the commands which just read it (those mentioned above) do not, so it is possible to they may hit a window during which it is corrupt. This is not a problem in practice since mount and umount are not frequent operations.

Name

multiuser - Bring the system up multiuser

Syntax

multiuser

Description

Multiuser prompts the user to set the current system date and time, and then brings the system up in multiuser mode.

First, multiuser displays the current system date and time and asks the user to confirm or change the date and then the time. Confirmation is done by entering Return. The format for entering the date is "yymmdd." Time is entered as a 24-hour clock in the form "hhmm."

See Also

date(C)

Name

mv - Move or rename files and directories

Syntax

mv filel file2

mv file ... directory

Description

Mv moves (changes the name of) filel to file2.

If file2 already exists, it is removed before filel is moved. If file2 has a mode which forbids writing, mv prints the mode (see chmod(F)) and reads the standard input to obtain a line; if the line begins with y, the move takes place; is not, mv exits.

In the second form, one or more files are moved to the directory with their original file-names.

Mv refuses to move a file onto itself.

See Also

cp(C), chmod(C)

Notes

If filel and file2 lie on different file systems, mv must copy the file and delete the original. In this case the owner name becomes that of the copying process and any linking relationship with other files is lost.

Name

null – The null file.

Description

Data written on a null special file is discarded.

Reads from a null special file always return 0 bytes.

Files

/dev/null

Name

passwd - Change login password

Syntax

passwd [name]

Description

This command changes (or installs) a password associated with the user name (your own name by default).

The program prompts for the old password and then for the new one. The user must supply both. The new password must be typed twice, to forestall mistakes.

New passwords must be at least four characters long if they use a sufficiently rich alphabet and at least six characters long if monocase. These rules are relaxed if you are insistent enough.

Only the owner of the name or the super-user may change a password; the owner must prove he knows the old password. Only the super-user can create a null password.

The password file is not changed if the new password is the same as the old password, or if the password has not "aged" sufficiently; see passwd(M)).

The minimum length of a legal password, and the minimum and maximum number of weeks used in password aging are specified in /etc/default/passwd by the variables PASSLENGTH, MINWEEKS and MAXWEEKS. For example, these variables might be set as follows:

```
PASSLENGTH=6
MINWEEKS=2
MAXWEEKS=6
```

FILES

/etc/default/passwd

/etc/passwd

See Also

login(C), passwd(M), crypt(S), default(M), pwadmin(C)

Name

passwd – The password file.

Description

Passwd contains the following information for each user:

- Login name
- Encrypted password
- Numerical user ID
- Numerical group ID
- Comment
- Initial working directory
- Program to use as shell

This is an ASCII file. Each field within each user's entry is separated from the next by a colon (:). The comment can contain any desired information. Each user is separated from the next by a newline. If the password field is null, no password is demanded; if the shell field is null, *sh*(C) is used.

This file resides in the directory /etc. Because the passwords are encrypted, the file has general read permission and can be used, for example, to map numerical user IDs to names.

The encrypted password consists of 13 characters chosen from a 64-character alphabet (., /, 0– 9, A– Z, a– z), except when the password is null, in which case the encrypted password is also null. Password aging is in effect for a particular user if his encrypted password in the password file is followed by a comma and a nonnull string of characters from the above alphabet. (Such a string must be introduced by the super-user.) The first character of the age denotes the maximum number of weeks for which a password is valid. A user who attempts to log in after his password has expired will be forced to supply a new one. The next character denotes the minimum period in weeks which must expire before the password may be changed. The remaining characters define the week (counted from the beginning of 1970) when the password was last changed. (A null string is equivalent to zero.) The first and second characters must have numerical values in the range 0– 63, where the dot (.) is equal to 0 and lowercase *z* is equal to 63. If the numerical value of both characters is 0, the user will be forced to change his password the next time he logs in. If the second character is greater than the first, only the super-user will be able to change the password.

Files

/etc/passwd

See Also

login(M), passwd(C), a64l(S), crypt(S), getpwent(S), group(M), pwadmin(C).

Name

/usr/bin/pconfig -- port configuration

Syntax

pconfig [-h | !]

Description

Pconfig is used for the maintenance of (tty) ports information. The pconfig utility allows you to associate physical devices such as CRTs, modems, and printers with specific ports on the computer. It also provides a mechanism for specifying the communication speed of these devices. For printers, pconfig provides a way to specify the parity and word length and then to test if the printer is properly configured.

This command may only be run by the system administrator (super user) -- it will abort if it is run by a user not having super-user status.

The following options are interpreted by pconfig:

-h Display the program's current version number and date, as well as a short description of the program's functions.

! Force execution of pconfig even if someone else is already running the program. This option may be used to override the "lock" which normally inhibits simultaneous execution of the program by more than one person. It is NOT recommended that more than one person use pconfig simultaneously since the resulting final state of the port configuration files may be erroneous.

Pconfig will issue the appropriate commands (e.g., setmode, tran, txset, and modem) to set up a port for immediate use. Also, it will add the appropriate commands to the /etc/rc file so that the ports will be configured automatically when the system starts up in multiuser mode.

The prompt at the bottom of the screen indicates that pconfig is awaiting input. The commands are:

c - change a port assignment

d - display all port assignments

h - display "help" information

q - update the system port assignment files, then terminate the program

r - remove a port assignment

t - test a printer

! - pass a command to XENIX for execution

All input to pconfig must be terminated with a carriage return or line feed.

Typing an interrupt (usually RUBOUT or DEL) will cause pconfig to immediately return to the top-level command interpreter.

Pconfig returns an exit code of one (1) if one or more changes were made to the configuration environment. It returns zero (Ø) if no changes were made. Thus pconfig's actions may be determined from a shell script.

Change allows the modification of a specified port. The user is prompted for the port he wishes to modify. He then is permitted to change the device type (terminal or printer), the port communication speed, the type of terminal to be attached to the port, and the printer number. Also, the user can specify an auxiliary printer or a modem if either is to be attached to the port.

Valid device types are "terminal", "printer", and "none". If there is no device plugged into a port, use "none".

Valid port names are displayed by pconfig when the display command is issued. Port names can be either hardware names (that correspond to the port names marked on the back of the computer) or software names (that correspond to the names used by XENIX). The hardware names are stored in file /etc/ports.info, and are names such as PORT 1, PORT 2, AND CRT. Software port names are stored in file /etc/ttys, and are names such as console, ttyl, and tty2.

If the device is a printer, pconfig will ask which printer number should be used. (Valid printer numbers are Ø through 9 and "default", which is the same as printer Ø). Also, pconfig will ask for the parity setting of the printer. The parity can be "even", "odd", or "none". If "none" is selected, then the word length (i.e. character size) is set to 8 bits; otherwise, the word length is set to 7-bits.

If the device is an Altos II terminal, it is possible to connect a printer to the auxiliary port on the back of the terminal. Pconfig will ask if such a printer is attached and, if so, what printer number should be used. (Valid printer numbers are Ø through 9 and "default", which is the same as printer Ø).

If the device is either a terminal or a printer, pconfig will also ask if a modem is attached.

Display shows the current status of the port assignment information. This display is updated by the change command. When a quit command is executed, this information becomes the final port assignment.

Help displays an informative screen telling about the conventions used by pconfig.

Quit exits the program. If any changes have been made to the port assignments, the operator is asked if the changes are correct, and if he answers "yes", the changes are installed .in the system. (The changes become effective immediately when pconfig exits.) If he answers "no", the operator is asked if he wants to exit anyway, and if he answers "yes" to that question, the program exits without making any changes.

Remove removes a port from use by the system. Normally, there will be no need to use this command. Currently, the only time that a port needs to be removed is when WorkNet is installed, in which case PORT 3 on an Altos 586 or 986 is disabled. (The WorkNet installation procedures automatically remove PORT 3.) The change command can be used in those rare instances when it is necessary to enable a removed port for use again.

Test allows a specified printer to be tested to check if the baud rate, parity, and word length settings are correct. It prints the alphabet in both upper and lower case, plus the numbers and some special characters. If the wrong data is printed, use the change command to try another combination of settings.

The exclamation point (!) escapes to the shell (see sh(C)). If no arguments are given, a shell is invoked that will continue until it receives an end-of-file (for example, <Control-D>). Then pconfig resumes. If arguments are present, a shell is invoked with the "-c" option and the arguments are passed along. Pconfig resumes immediately thereafter. If csh(C) is desired rather than sh(C), the command !csh should be used.

Any command which is not understood by pconfig causes an appropriate message to be displayed.

Pconfig does not distinguish between RETURN and LINE FEED. They may be used interchangeably.

Pconfig will immediately return to the top-level command interpreter upon receipt of an interrupt signal, such as the DEL, RUBOUT or BREAK key.

During the update procedure pconfig copies /etc/ttys, /etc/ttytype, /etc/ports.info, and /etc/rc to /etc/ottys, /etc/ottytype, /etc/oports.info, and /etc/orc respectively. Thus, if a mistake or disaster occurs during the use of this program, the user may recover the prior state of any or all of these files.

Files

/etc/ttys	software port name data file
/etc/ttytype	terminal type data file
/etc/ports.info	hardware port name data file
/etc/rc	commands to run when starting up multi-user
/etc/ottys	this file is a copy of /etc/ttys before any modifications are made
/etc/ottytype	this file is a copy of /etc/ttytype before any modifications are made
/etc/oports.info	this file is a copy of /etc/ports.info before any modifications are made
/etc/orc	this file is a copy of /etc/rc before any modifications are made
/etc/termcap	contains terminal attribute descriptions
/etc/pconfig.lock	lock file
/etc/pconfig.temp	temporary file
/etc/pconfig.temp1	temporary file
/etc/pconfig.temp2	temporary file

See Also

termcap(F), ttytype(F), ttys(F)

Diagnostics

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

Notes

Complete consistency checking among the /etc/ttys, /etc/ttytype, /etc/ports.info and /etc/termcap files is not done.

If pconfig is run in system maintenance mode (i.e., single-user mode), it may leave extraneous setmode commands running when multiuser mode is started.

Pconfig does not check for a printer currently in use.

Name

printers - Print spooler configuration file

Description

The printer spooler facility (lpr) enables the user to print a specified list of files on one or several line printers. Additionally, a printer on a machine connected to WORKNET, should be shared by other machines on the same net. Such printers may need to have an arbitrary set of terminal modes set on its device, for tab expansion, baud rate, etc. The system printer configuration file (/etc/printers) consists of lines of printer configuration information, which include WORKNET machine names, ttytypes, device names, and tty modes.

```
LP[P]:NAME:TTYTYPE:[NETNAME]:[TTYMODES]
```

Fields are separated by ":" and may not contain spaces between the colon separators and field values. The length of this line may not exceed 128 characters. Comments are permitted in the configuration file. A comment line begins with "#" in the first column. Any fields surrounded by "[]" are optional, although their colon separators are not.

The first field, LP[P], is the printer device select. Allowable values for P are null or a single numeric digit. This value is used to specify one of several printers.

The second field, NAME, is a tag by which a particular configuration line can be selected. Allowable values are alphanumeric strings, which do not contain the ":" character.

TTYTYPE is not used by the print spooler, but exists for the convenience of word processing programs that derive printer control sequences from /etc/termcap (or similar database).

The NETNAME field may be null, which indicates that spooling is to take place on the requestor's machine. Other values are net machine names. The print spooler uses this name to do remote printing.

The TTYMODES list is a sequence of whitespace delimited tty mode specifications, such as would be supplied to stty(C).

Example:

```
# a printer configuration file
lp:calcite:NEC3510:gateway:
lp:galena:Oki93::-tabs 1200 nl
```

```
lp0:obsidian:I9:Marketing:tabs 9600 nl
lp1:feldspar:epson::nl tabs 9600
lp2:mica:TI810:Finance:9600 -tabs
```

A synopsis of the above lines:

The first line uses the /dev/lp printer on the machine named "gateway". The printer type is "NEC3510" and no tty modes are to be set on that printer. This line may be selected by specifying "calcite" to lpr.

The next line specifies the /dev/lp printer on the user's local machine (note the null NETNAME field), is type Oki93 and sets tab expansion (-tabs), 1200 baud operation, and no linefeed to cr-lf expansion. This line is selected with the name "galena".

The third line requests /dev/lp0, is on the Marketing machine, runs the printer at 9600 baud, etc., is type I9, and is selected by the name "obsidian".

The last two lines use /dev/lp1 on the local machine, and /dev/lp2 on the Finance machine.

Files

/etc/printers	printer mode control file

See Also

lpr(C), lpd(M), tty(M), lp(M)

Name

profile – Sets up an environment at login time.

Description

The optional file .profile permits automatic execution of commands whenever a user logs in. The file is generally used to personalize a user's work environment by setting exported environment variables and terminal mode (see environ(C)).

When a user logs in, the user's login shell looks for .profile in the login directory. If found, the shell executes the commands in the file before beginning the session. The commands in the file must have the same format as if typed at the keyboard. Any line beginning with the number sign (#) is considered a comment and is ignored. The following is an example of a typical file:

```
# Tell me when new mail comes in
MAIL=/usr/mail/myname
# Add my /bin directory to the shell search sequence
PATH=$PATH:$HOME/bin
# Make some environment variables global
export MAIL PATH TERM
# Set file creation mask
umask 22
```

Note that the file /etc/profile is a system-wide profile that, if it exists, is executed for every user before the user's .profile is executed.

Files

$HOME/.profile
/etc/profile

See Also

env(C), login(M), mail(C), sh(C), stty(C), su(C), environ(M)

Name

ps - Reports process status.

Syntax

ps [options]

Description

Ps prints certain information about active processes. Without options, information is printed about processes associated with the current terminal. Otherwise, the information that is displayed is controlled by the following options:

- e — Prints information about all processes.

- d — Prints information about all processes, except process group leaders.

- a — Prints information about all processes, except process group leaders and processes not associated with a terminal.

- f — Generates a full listing. (Normally, a short listing containing only process ID, terminal ("tty") identifier, cumulative execution time, and the command name is printed.) See below for meaning of columns in a full listing.

- l — Generates a long listing. See below.

- c corefile — Uses the file corefile in place of /dev/kmem.

- s swapdev — Uses the file swapdev in place of /dev/swap. This is useful when examining a corefile.

- n namelist — The argument is taken as the name of an alternate namelist (/xenix is the default).

- t tlist — Restricts listing to data about the processes associated with the terminals given in tlist, where tlist can be in one of two forms: a list of terminal identifiers separated from one another by a comma, or a list of terminal identifiers enclosed in double quotes and separated from one another by a comma and/or one or more spaces.

- p plist Restricts listing to data about processes whose process ID numbers are given in plist, where plist is in the same format as tlist.

- u ulist Restricts listing to data about processes whose user ID numbers or login names are given in ulist, where ulist is in the same format as tlist. In the listing, the numerical user ID is printed unless the - f option is used, in which case the login name is printed.

- g glist Restricts listing to data about processes whose process groups are given in glist, where glist is a list of process group leaders and is in the same format as tlist.

The column headings and the meaning of the columns in a ps listing are given below; the letters f and l indicate the option (full or long) that causes the corresponding heading to appear; all means that the heading always appears. Note that these two options only determine what information is provided for a process; they do not determine which processes will be listed.

F (l) A status word consisting of flags associated with the process. Each flag is associated with a bit in the status word. These flags are added to form a single octal number. Process flag bits and their meanings are:

01	in core;
02	system process;
04	locked in core (e.g., for physical I/O);
10	being swapped;
20	being traced by another process.

S (l) The state of the process;

O	non-existent;
S	sleeping;
W	waiting;
R	running;
I	intermediate;
Z	terminated;
T	stopped.

UID (f,l) The user ID number of the process owner; the login name is printed under the - f option.

PID	(all)	The process ID of the process; it is possible to kill a process if you know this datum.
PPID	(f,l)	The process ID of the parent process.
C	(f,l)	Process utilization for scheduling.
STIME	(f)	Starting time of the process.
PRI	(l)	The priority of the process; higher numbers mean lower priority.
NI	(l)	Nice value; used in priority computation.
ADDR	(l)	The memory address of the process, if resident; otherwise, the disk address.
SZ	(l)	The size in blocks of the core image of the process.
WCHAN	(l)	The event for which the process is waiting or sleeping; if blank, the process is running.
TTY	(all)	The controlling terminal for the process.
TIME	(all)	The cumulative executive time for the process.
CMD	(all)	The command name; the full command name and its arguments are printed under the - f option.

A process that has exited and has a parent, but has not yet been waited for by the parent, is marked <defunct>.

Under the - f option, ps tries to determine the command name and arguments given when the process was created by examining memory or the swap area. Failing this, the command name, as it would appear without the - f option, is printed in square brackets.

Files

/xenix	system namelist
/dev/mem	memory
/dev	searched to find swap device and terminal ("tty") names.

See Also

kill(C), nice(C)

Notes

Things can change while ps is running; the picture it gives is only a close approximation to reality.

Some data printed for defunct processes are irrelevant.

Name

pwd - Prints working directory name

Syntax

pwd

Description

Pwd prints the pathname of the working (current) directory.

See Also

cd(C)

Diagnostics

"Cannot open .." and "Read error in .." indicate possible file system trouble. In such cases, see the XENIX Operations Guide for information on fixing the file system.

Name

reset - Reset the teletype bits to a sensible state

Syntax

reset

Description

Reset sets the teletype bits to 'soft-copy terminal standard mode' with the erase character set to control-h and the kill character to 'Ø'. Reset is most useful when you bomb out in raw mode.

See Also

stty(C), stty(5), gtty(5)

Notes

If you are in a funny state you may well have to type "reset" followed by linefeed (control-j if there is no such key).

Name

restor - Invokes incremental file system restorer.

Syntax

restor key [arguments]

Description

Restor is used to read archive media backed up with the dump command. The key specifies what is to be done. Key is one of the characters rRxt, optionally combined with f.

f Uses the first argument as the name of the archive instead of the default.

r,R The archive is read and loaded into the file system specified in argument. This should not be done lightly (see below). If the key is R, restor asks which archive of a multivolume set to start on. This allows restor to be interrupted and then restarted (an fsck must be done before the restart).

x Each file on the archive named by an argument is extracted. The filename has all "mount" prefixes removed; for example, if /usr is a mounted file system, /usr/bin/lpr is named /bin/lpr on the archive. The extracted file is placed in a file with a numeric name supplied by restor (actually the inode number). In order to keep the amount of archive read to a minimum, the following procedure is recommended:

1. Mount volume 1 of the set of backup archives.

2. Type the restor command.

3. Restor will announce whether or not it found the files, give the numeric name that it will assign to the file, and in the case of a tape, rewind to the start of the archive.

4. It then asks you to "mount the desired tape volume". Type the number of the volume you choose. On a multivolume backup the recommended procedure is to mount the last through the first volumes, in that order. Restor checks to see if any of the requested files are on the mounted archive (or a later archive-thus the reverse order). If the requested files are not there, restor doesn't read through the tape. If you are

working with a single-volume backup or if the number of files being restored is large, respond to the query with 1 and restor will read the archives in sequential order.

t Prints the date the archive was written and the date the file system was backed up.

The r option should only be used to restore a complete backup archive onto a clear file system, or to restore an incremental backup archive onto a file system so created. Thus:

```
/etc/mkfs /dev/hd1 10000
restor r /dev/hd1
```

is a typical sequence to restore a complete backup. Another restor can be done to get an incremental backup in on top of this.

A dump followed by a mkfs and a restor is used to change the size of a file system.

Files

rst*	Temporary files
/etc/default/dump	Name of default archive device

The default archive unit varies with installation.

See Also

dump(C), fsck(C), mkfs(C)

Diagnostics

There are various diagnostics involved with reading the archive and writing the disk. There are also diagnostics if the i-list or the free list of the file system is not large enough to hold the dump.

If the dump extends over more than one disk or tape, it may ask you to change disks or tapes. Reply with a newline when the next unit has been mounted.

Notes

It is not possible to successfully restor an entire active root file system.

Name

rm, rmdir - Remove files or directories

Syntax

rm [-fri] file ...

rmdir dir ...

Description

Rm removes the entries for one or more files from a directory. If an entry was the last link to the file, the file is destroyed. Removal of a file requires write permission in its directory, but neither read nor write permission on the file itself.

If a file has no write permission and the standard input is a terminal, its permissions are printed and a line is read from the standard input. If that line begins with 'y' the file is deleted, otherwise the file remains. No questions are asked when the -f (force) option is given.

If a designated file is a directory, an error comment is printed unless the optional argument -r has been used. In that case, rm recursively deletes the entire contents of the specified directory, and the directory itself.

If the -i (interactive) option is in effect, rm asks whether to delete each file, and, under -f, whether to examine each directory.

Rmdir removes entries for the named directories, which must be empty.

Diagnostics

Generally self-explanatory. It is forbidden to remove the file '..' merely to avoid the antisocial consequences of inadvertently doing something like 'rm -r.*'.

It is also forbidden to remove the root directory of a given file system.

Name

sed - Invokes the stream editor.

Syntax

sed [-n] [-e script] [-f sfile] [files]

Description

Sed copies the named files (standard input default) to the standard output, edited according to a script of commands. The -f option causes the script to be taken from file sfile; these options accumulate. If there is just one -e option and no -f options, the flag -e may be omitted. The -n option suppresses the default output. A script consists of editing commands, one per line, of the following form:

[address [, address]] function [arguments]

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

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

An address is either a decimal number that counts input lines cumulatively across files, a $ that addresses the last line of input, or a context address, i.e., a /regular expression/ in the style of ed(C) modified as follows:

- In a context address, the construction \?regular expression?, where ? is any character, is identical to /regular expression/. Note that in the context address \xabc\xdefx, the second x stands for itself, so that the regular expression is abcxdef.

- The escape sequence \n matches a newline embedded in the pattern space.

- A period . matches any character except the terminal newline of the pattern space.

- A command line with no addresses selects every pattern space.

- A command line with one address selects each pattern space that matches the address.

- A command line with two addresses selects the inclusive range from the first pattern space that matches the first address through the next pattern space that matches the second. (If the second address is a number less than or equal to the line number first selected, only one line is selected.) Thereafter the process is repeated, looking again for the first address.

Editing commands can be applied only to nonselected pattern spaces by use of the negation function ! (below).

In the following list of functions the maximum number of permissible addresses for each function is indicated in parentheses.

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

(1)a\
text — Appends text, placing it on the output before reading the next input line.

(2)b label — Branches to the : command bearing the label. If label is empty, branches to the end of the script.

(2)c\
text — Changes text by deleting the pattern space and then appending text. With 0 or 1 address or at the end of a 2-address range, places text on the output and starts the next cycle.

(2)d — Deletes the pattern space and starts the next cycle.

(2)D — Deletes the initial segment of the pattern space through the first newline and starts the next cycle.

(2)g — Replaces the contents of the pattern space with the contents of the hold space.

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

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

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

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

(2)l Lists the pattern space on the standard output with nonprinting characters spelled in two-digit ASCII and long lines folded.

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

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

(2)p Prints (copies) the pattern space on the standard output.

(2)P Prints (copies) the initial segment of the pattern space through the first newline to the standard output.

(1)q Quits sed by branching to the end of the script. No new cycle is started.

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

(2)s/regular expression/replacement/flags
Substitutes the replacement string for instances of the regular expression in the pattern space. Any character may be used instead of /. For a more detailed description see ed(C). Flags is zero or more of:

g Globally substitutes for all nonoverlapping instances of the regular expression rather than just the first one.

p Prints the pattern space if a replacement was made.

w wfile
Writes the pattern space to wfile if a replacement was made.

(2) t label — Branches to the colon (:) command bearing label if any substitutions have been made since the most recent reading of an input line or execution of a t command. If label is empty, t branches to the end of the script.

(2) w wfile — Writes the pattern space to wfile.

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

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

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

(0):label — This command does nothing; it bears a label for b and t commands to branch to.

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

(2){ — Executes the following commands through a matching } only when the pattern space is selected.

(0) — An empty command is ignored.

See Also

awk(C), ed(C), grep(C)
The XENIX Text Processing Guide

Notes

This command is more fully documented in the XENIX Text Processing Guide.

Name

setmnt - Establishes /etc/mnttab table.

Syntax

/etc/setmnt

Description

Setmnt creates the /etc/mnttab table (see mnttab(F)), which is needed for both the mount(C) and umount(C) commands. Setmnt reads the standard input and creates a mnttab entry for each line. Input lines have the format:

filesys node

where filesys is the name of the file system's special file (e.g., "hd0") and node is the root name of that file system. Thus filesys and node become the first two strings in the mnttab(F) entry.

Files

/etc/mnttab

See Also

mnttab(F)

Notes

If filesys or node are longer than 128 characters errors can occur.

Setmnt silently enforces an upper limit on the maximum number of mnttab entries.

Setmnt is normally invoked by /etc/rc when the system boots up.

Name

setmode - Printer modes utility

Syntax

setmode dev modes

Description

Setmode sets tty modes (see tty(M)) for tty ports which are used for serial printers, i.e., not logged in (see disable(C)). The primary use of this program is to set up serial baud, tab expansion, and newline actions for programs that do output directly to a serial port.

Setmode takes a list of tty modes from its command line, does an stty(C) on the indicated device, and sleeps forever. This has the effect of keeping the device open with the desired modes. Setmode should be invoked once for each printer device that is to be used by a program which does not use the print spooler, lpr(C). The lpr(C) program uses a different technique for setting modes on the serial ports it uses (see lpr(C)). The /etc/rc file is a good place to invoke setmode since it ensures that setmode will be run every time the system is brought up.

Setmode must be invoked with at least two arguments, which are the name of the device (special file) and at least one tty mode. Setmode will complain if it cannot fork, exec stty, or it has a strange tty mode handed to it.

Files

/usr/bin/setmode	the setmode program
/dev/lp?	printer devices

See Also

lpr(C), stty(C), disable(C), tty

Name

sh - Invokes the shell command interpreter.

Syntax

sh [- ceiknrstuvx] [args]

Description

The shell is the standard command programming language that executes commands read from a terminal or a file. See Invocation below for the meaning of arguments to the shell.

Commands

A simple command is a sequence of nonblank words separated by blanks (a blank is a tab or a space). The first word specifies the name of the command to be executed. Except as specified below, the remaining words are passed as arguments to the invoked command. The command name is passed as argument Ø (see exec(S)). The value of a simple command is its exit status if it terminates normally, or (octal) 2ØØ+ status if it terminates abnormally, i.e., if the failure produces a core file. See signal(S) for a list of status values.

A pipeline is a sequence of one or more commands separated by a vertical bar (|). The caret (^) also has the same effect.) The standard output of each command but the last is connected by a pipe(S) to the standard input of the next command. Each command is run as a separate process; the shell waits for the last command to terminate.

A list is a sequence of one or more pipelines separated by ;, &, &&, or || and optionally terminated by ; or &. Of these four symbols, ; and & have equal precedence, which is lower than that of && and ||. The symbols && and || also have equal precedence. A semicolon (;) causes sequential execution of the preceding pipeline; an ampersand (&) causes asynchronous execution of the preceding pipeline (i.e., the shell does not wait for that pipeline to finish). The symbol && (||) causes the list following it to be executed only if the preceding pipeline returns a zero (nonzero) exit status. An arbitrary number of newlines may appear in a list, instead of semicolons, to delimit commands.

A command is either a simple command or one of the following commands. Unless otherwise stated, the value returned by a command is that of the last simple command executed in the command:

for name [in word ...] do list done

Each time a for command is executed, name is set to the next word taken from the in word list. If in word is omitted, then the for command executes the do list once for each positional parameter that is set (see Parameter Substitution below). Execution ends when there are no more words in the list.

case word in [pattern [|pattern] ...) list ;;] ... esac

A case command executes the list associated with the first pattern that matches word. The form of the patterns is the same as that used for filename generation (see Filename Generation below).

if list then list [elif list then list] ... [else list] fi

The list following if is executed and, if it returns a zero exit status, the list following the first then is executed. Otherwise, the list following elif is executed and, if its value is zero, the list following the next then is executed. Failing that, the else list is executed. If no else list or then list is executed, then the if command returns a zero exit status.

while list do list done

A while command repeatedly executes the while list and, if the exit status of the last command in the list is zero, executes the do list; otherwise the loop terminates. If no commands in the do list are executed, then the while command returns a zero exit status; until may be used in place of while to negate the loop termination test.

(list)

Executes list in a subshell.

{list;}

list is simply executed.

The following words are only recognized as the first word of a command and when not quoted:

if then else elif fi case esac for while until do done { }

Comments

A word beginning with # causes that word and all the following characters up to a newline to be ignored.

Command Substitution

The standard output from a command enclosed in a pair of grave accents (``) may be used as part or all of a word; trailing newlines are removed.

Parameter Substitution

The character $ is used to introduce substitutable parameters. Positional parameters may be assigned values by set. Variables may be set by writing:

name==value [name==value] ...

Pattern-matching is not performed on value.

${parameter}

A parameter is a sequence of letters, digits, or underscores (a name), a digit, or any of the characters *, @, #, ?, -, $, and !. The value, if any, of the parameter is substituted. The braces are required only when parameter is followed by a letter, digit, or underscore that is not to be interpreted as part of its name. A name must begin with a letter or underscore. If parameter is a digit, then it is a positional parameter. If parameter is * or @, then all the positional parameters, starting with $1, are substituted (separated by spaces). Parameter $0 is set from argument zero when the shell is invoked.

${parameter:- word}

If parameter is set and is nonnull then substitute its value; otherwise substitute word.

${parameter:==word}

If parameter is not set or is null, then set it to word; the value of the parameter is then substituted. Positional parameters may not be assigned to in this way.

${parameter:?word}

If parameter is set and is nonnull then substitute its value; otherwise, print word and exit from the shell. If word is omitted, then the message "parameter null or not set" is printed.

${parameter:+word}

If parameter is set and is nonnull then substitute word; otherwise substitute nothing.

In the above, word is not evaluated unless it is to be used as the substituted string, so that in the following example, pwd is executed only if d is not set or is null:

```
echo ${d:- `pwd`}
```

If the colon (:) is omitted from the above expressions, then the shell only checks whether parameter is set.

The following parameters are automatically set by the shell:

\# The number of positional parameters in decimal

\- Flags supplied to the shell on invocation or by the set command

? The decimal value returned by the last synchronously executed command

$ The process number of this shell

! The process number of the last background command invoked

The following parameters are used by the shell:

HOME
The default argument (home directory) for the cd command

PATH
The search path for commands (see Execution below)

MAIL
If this variable is set to the name of a mail file, then the shell informs the user of the arrival of mail in the specified file

PS1 Primary prompt string, by default "$"

PS2 Secondary prompt string, by default ">"

IFS Internal field separators, normally space, tab, and newline

The shell gives default values to PATH, PS1, PS2, and IFS, while HOME and MAIL are not set at all by the shell (although HOME is set by login(M)).

Blank Interpretation

After parameter and command substitution, the results of substitution are scanned for internal field separator characters (those found in IFS) and split into distinct arguments where such characters are found. Explicit null arguments (** or '') are retained. Implicit null arguments (those resulting from parameters that have no values) are removed.

Filename Generation

Following substitution, each command word is scanned for the characters *, ?, and [. If one of these characters appears then the word is regarded as a pattern. The word is replaced with alphabetically sorted filenames that match the pattern. If no filename is found that matches the pattern, then the word is left unchanged. The character . at the start of a filename or immediately following a /, as well as the character / itself, must be matched explicitly. These characters and their matching patterns are:

* Matches any string, including the null string.

? Matches any single character.

[...]
Matches any one of the enclosed characters. A pair of characters separated by - matches any character lexically between the pair, inclusive.

Quoting

The following characters have a special meaning to the shell and cause termination of a word unless quoted:

; & ( ) | ^ < > newline space tab

A character may be quoted (i.e., made to stand for itself) by preceding it with a \. The pair \newline is ignored. All characters enclosed between a pair of single quotation marks (''), except a single quotation mark, are quoted. Inside double quotation marks (""), parameter and command

substitution occurs and \ quotes the characters \, `, :, and $. "$*" is equivalent to "$1 $2 ...", whereas "$@" is equivalent to "$1" "$2" ...

Prompting

When used immediately, the shell prompts with the value of PS1 before reading a command. If at any time a newline is typed and further input is needed to complete a command, then the secondary prompt (i.e., the value of PS2) is issued.

Input/Output

Before a command is executed, its input and output may be redirected using a special notation interpreted by the shell. The following may appear anywhere in a simple-command or may precede or follow a command. They are not passed on to the invoked command; substitution occurs before word or digit is used:

<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 then it is created; otherwise, it is truncated to zero length.

>>word — Use file word as standard output. If the file exists then output is appended to it (by first seeking to the end-of-file); otherwise, the file is created.

<<[-]word — The shell input is read up to a line that is the same as word, or to an end-of-file. The resulting document becomes the standard input. If any character of word is quoted, then no interpretation is placed upon the characters of the document; otherwise, parameter and command substitution occurs, (unescaped) \newline is ignored, and \ must be used to quote the characters \, $, `, and the first character of word. If .. is appended to <<, then all leading tabs are stripped from word and from the document.

<&digit — The standard input is duplicated from file descriptor digit (see dup(S)). Similarly for the standard output using >.

<&- — The standard input is closed. Similarly for the standard output using >.

If one of the above is preceded by a digit, then the file descriptor created is that specified by the digit (instead of the default Ø or 1). For example:

```
... 2>&1
```

creates file descriptor 2 that is a duplicate of file descriptor 1.

If a command is followed by & then the default standard input for the command is the empty file /dev/null. Otherwise, the environment for the execution of a command contains the file descriptors of the invoking shell as modified by input/output specifications.

Environment

The environment (see environ(M)) is a list of name-value pairs that is passed to an executed program in the same way as a normal argument list. The shell interacts with the environment in several ways. On invocation, the shell scans the environment and creates a parameter for each name found, giving it the corresponding value. Executed commands inherit the same environment. If the user modifies the values of these parameters or creates new ones, none of these affects the environment unless the export command is used to bind the shell's parameter to the environment. The environment seen by any executed command is thus composed of any unmodified name-value pairs originally inherited by the shell, plus any modifications or additions, all of which must be noted in export commands.

The environment for any simple command may be augmented by prefixing it with one or more assignments to parameters. Thus:

```
TERM==45Ø cmd args
```

and

```
(export TERM; TERM==45Ø; cmd args)
```

are equivalent (as far as the above execution of cmd is concerned).

If the - k flag is set, all keyword arguments are placed in the environment, even if they occur after the command name.

Signals

The INTERRUPT and QUIT signals for an invoked command are ignored if the command is followed by &; otherwise signals

have the values inherited by the shell from its parent, with the exception of signal 11 (but see also the trap command below).

Execution

Each time a command is executed, the above substitutions are carried out. Except for the Special Commands listed below, a new process is created and an attempt is made to execute the command via exec(S).

The shell parameter PATH defines the search path for the directory containing the command. Alternative directory names are separated by a colon (:). The default path is :/bin:/usr/bin (specifying the current directory, /bin, and /usr/bin, in that order). Note that the current directory is specified by a null pathname, which can appear immediately after the equal sign or between the colon delimiters anywhere else in the path list. If the command name contains a / then the search path is not used. Otherwise, each directory in the path is searched for an executable file. If the file has execute permission but is not an a.out file, it is assumed to be a file containing shell commands. A subshell (i.e., a separate process) is spawned to read it. A parenthesized command is also executed in a subshell.

Special Commands

The following commands are executed in the shell process and, except as specified, no input/output redirection is permitted for such commands:

: No effect; the command does nothing. A zero exit code is returned.

. file
Reads and executes commands from file and returns. The search path specified by PATH is used to find the directory containing file.

break [n]
Exits from the enclosing for or while loop, if any. If n is specified then breaks n levels.

continue [n]
Resumes the next iteration of the enclosing for or while loop. If n is specified then resumes at the n-th enclosing loop.

cd [arg]
Changes the current directory to arg. The shell parameter HOME is the default arg.

eval [arg ...]
The arguments are read as input to the shell and the resulting command(s) executed.

exec [arg ...]
The command specified by the arguments is executed in place of this shell without creating a new process. Input/output arguments may appear and, if no other arguments are given, cause the shell input/output to be modified.

exit [n]
Causes a shell to exit with the exit status specified by n. If n is omitted then the exit status is that of the last command executed (an end-of-file will also cause the shell to exit.)

export [name ...]
The given names are marked for automatic export to the environment of subsequently executed commands. If no arguments are given, a list of all names that are exported in this shell is printed.

newgrp [arg ...]
Equivalent to exec newgrp arg ...

read [name ...]
One line is read from the standard input and the first word is assigned to the first name, the second word to the second name, etc., with leftover words assigned to the last name. The return code is 0 unless an end-of-file is encountered.

readonly [name ...]
The given names are marked readonly and the values of these names may not be changed by subsequent assignment. If no arguments are given, then a list of all readonly names is printed.

set [- ekntuvx [arg ...]]

- e If the shell is noninteractive, exits immediately if a command exits with a nonzero exit status.

- k Places all keyword arguments in the environment for a command, not just those that precede the command name.

- n Reads commands but does not execute them.

- t Exits after reading and executing one command. Intended for use by C programs only; not useful interactively.

- u Treats unset variables as an error when substituting.

- v Prints shell input lines as they are read.

- x Prints commands and their arguments as they are executed.

- - Does not change any of the flags; useful in setting $1 to -.

Using + rather than - causes these flags to be turned off. These flags can also be used upon invocation of the shell. The current set of flags may be found in $-. The remaining arguments are positional parameters and are assigned, in order, to $1, $2, ... If no arguments are given then the values of all names are printed.

shift
The positional parameters from $2 ... are renamed $1 ....

test
Evaluates conditional expressions. See test(C) for usage and description.

times
Prints the accumulated user and system times for processes run from the shell.

trap [arg] [n] ...
arg is a command to be read and executed when the shell receives signal(s) n. (Note that arg is scanned once when the trap is set and once when the trap is taken.) Trap commands are executed in order of signal number. The highest signal number allowed is 16. Any attempt to set a trap on a signal that was ignored on entry to the current shell is ineffective. An attempt to trap on signal 11 (memory fault) produces an error. If arg is absent then all trap(s) n are reset to their original values. If arg is the null string then this signal is ignored by the shell and by the commands it invokes. If n is 0 then the command arg is executed on exit from the shell. The trap command with no arguments prints a list of commands associated with each signal number.

umask [ooo]
The user file-creation mask is set to the octal number ooo where o is an octal digit (see umask(C)). If ooo is omitted, the current value of the mask is printed.

wait
Waits for all child processes to terminate, and reports the termination status. If n is not given than all currently active child processes are waited for. The return code from this command is always 0.

Invocation

If the shell is invoked through exec(S) and the first character of argument 0 is - , commands are initially read from /etc/profile and then from $HOME/.profile, if such files exist. Thereafter, commands are read as described below, which is also the cause when the shell is invoked as /bin/sh. The flags below are interpreted by the shell on invocation only; note that unless the - c or - s flag is specified, the first argument is assumed to be the name of a file containing commands, and the remaining arguments are passed as positional parameters to that command file:

- c string — If the - c flag is present then commands are read from string.

- s — If the - s flag is present or if no arguments remain then commands are read from the standard input. Any remaining arguments specify the positional parameters. Shell output is written to file descriptor 2.

- i — If the - i flag is present or if the shell input and output are attached to a terminal, then this shell is interactive. In this case TERMINATE is ignored (so that kill 0 does not kill an interactive shell) and INTERRUPT is caught and ignored (so that wait is interruptible). In all cases, QUIT is ignored by the shell.

- r — If the - r flag is present the shell is a restricted shell (see rsh(C)).

The remaining flags and arguments are described under the set command above.

Exit Status

Errors detected by the shell, such as syntax errors, cause the shell to return a nonzero exit status. If the shell is being used noninteractively then execution of the shell file is abandoned. Otherwise, the shell returns the exit status of the last command executed (see also the exit command above).

Files

/etc/profile

$HOME/.profile

/tmp/sh*

/dev/null

See Also

cd(C), env(C), login(M), newgrp(C), rsh(C), test(C), umask(C), dup(S), exec(S), fork(S), pipe(S), signal(S), umask(S), wait(S), a.out(F), profile(M), environ(M)

Notes

The command readonly (without arguments) produces the same output as the command export.

If << is used to provide standard input to an asynchronous process invoked by &, the shell gets mixed up about naming the input document; a garbage file /tmp/sh* is created and the shell complains about not being able to find that file by another name.

Name

shutdown - Terminates all processing.

Syntax

/etc/shutdown [time] [su]

Description

Shutdown is part of the XENIX operation procedures. Its primary function is to terminate all currently running processes in an orderly and cautious manner. The time argument is the number of minutes before a shutdown will occur; default is five minutes. The optional su argument lets the user go single-user, without completely shutting down the system. However, the system is shut down for multiuser use. Shutdown goes through the following steps. First, all users logged on the system are notified to log off the system by a broadcasted message. All file system super-blocks are updated before the system is stopped (see sync(C)). This must be done before rebooting the system, to insure file system integrity.

Files

/etc/reebooter

See Also

sync(C), umount(C), wall(C)

Diagnostics

The most common error diagnostic that will occur is device busy. This diagnostic appears when a particular file system could not be unmounted. See umount(C).

Notes

Once shutdown has been invoked it must be allowed to run to completion and must not be interrupted by pressing BREAK or DEL.

Name

sizefs - Determine the size of a logical device from the layout information associated with a hard disk.

Syntax

sizefs layout-file logical-device-number

Description

Sizefs prints on the standard output the size in blocks of the specified area on the disk. It gets its information out of the structure created by layout(C). Its most common use is in shell scripts for creating a file system on the hard disk, where its output is used as an argument to mkfs(C).

See Also

layout(C), map(C), mkfs(C)

Name

sleep - Suspend execution for an interval

Syntax

sleep time

Description

Sleep suspends execution for time seconds. It is used to execute a command after a certain amount of time as in:

```
(sleep 105; command)&
```

or to execute a command every so often, as in:

```
while true
do
     command
     sleep 37
done
```

See Also

alarm(S), sleep(S)

Notes

Time must be less than 65536 seconds.

Name

sort - Sorts and merges files.

Syntax

sort [-cmubdfinrtx] [+posl [-pos2]] ... [-o output] [files]

Description

Sort merges and sorts lines from all named files and writes the result on the standard output. A dash (-) may appear as a file in the files argument signifying the standard input. If no input files are named, the standard input is sorted.

The default sort key is an entire line. Default ordering is lexicographic by bytes in machine collating sequence. The ordering is affected globally by the following options, one or more of which may appear.

-b Ignores leading blanks (spaces and tabs) in field comparisons.

-d "Dictionary" order: only letters, digits and blanks are significant in comparisons.

-f Folds uppercase letters onto lowercase.

-i Ignores characters outside the ASCII octal range 040-0176 in non-numeric comparisons.

-n An initial numeric string, consisting of optional blanks, optional minus sign, and zero or more digits with optional decimal point, is sorted by arithmetic value. Option n implies option b.

-r Reverses the sense of comparisons.

tx "Tab character" separating fields is x.

The notation +posl -pos2 restricts a sort key to a field beginning at posl and ending just before pos2. Posl and pos2 each have the form m.n, optionally followed by one or more of the flags bdfinr, where m tells a number of fields to skip from the beginning of the line and n tells a number of characters to skip further. If any flags are present they override all the global ordering options for this key. If the b option is in effect n is counted from the first nonblank in the field; b is attached independently to pos2. A missing .n means .0; a missing -pos2 means the end of the

line. Under the -tx option, fields are strings separated by x; otherwise fields are nonempty nonblank strings separated by blanks.

When there are multiple sort keys, later keys are compared only after all earlier keys compare equal. Lines that otherwise compare equal are ordered with all bytes significant. Very long lines are silently truncated.

These option arguments are also understood:

-c Checks that the input file is stored according to the ordering rules; gives no output unless the file is out of sort.

-m Merges only, the input files are already sorted.

-u Suppresses all but one in each set of duplicated lines. Ignored bytes and bytes outside keys do not participate in this comparison.

-o The next argument is the name of an output file to use instead of the standard output. This file may be the same as one of the inputs.

Examples

The following prints in alphabetical order all the unique spellings in a list of words (capitalized words differ from uncapitalized):

```
sort -u +0f +0 list
```

The following prints the password file (passwd(F)) sorted by user ID (the third colon-separated field):

```
sort -t: +2n /etc/passwd
```

The following prints the first instance of each month in an already sorted file of (month-day) entries (the options -um with just one input file make the choice of a unique representative from a set of equal lines predictable):

```
sort -um +0 -1 dates
```

Files

/usr/tmp/stm???

See Also

comm(C), join(C), uniq(C)

Diagnostics

Comments and exits with nonzero status for various trouble conditions and for disorder discovered under option -c.

Name

stty - Sets the options for a terminal.

Syntax

stty [-a] [-g] [options]

Diagnostics

Stty sets certain terminal I/O options for the device that is the current standard input; without arguments, it reports the setting of certain options; with the -a option, it reports all of the option settings; with the -g option, it reports current settings in a form that can be used as an argument to another stty command. Detailed information about the modes listed in the first five groups below may be found in tty(M). Options in the last group are implemented using options in the previous groups. The options are selected from the following:

Control Modes

parenb (-parenb)
Enables (disables) parity generation and detection.

parodd (-parodd)
Selects odd (even) parity.

cs5 cs6 cs7 cs8
Selects character size (see tty(M)).

0 Hangs up phone line immediately.

50 75 110 134 150 200 300 600 1200 1800 2400 4800 9600 exta extb
Sets terminal baud rate to the number given, if possible.

hupcl (-hupcl)
Hangs up (does not hang) phone connection on last close.

hup (-hup)
Same as hupcl (-hupcl).

cstopb (-cstopb)
Uses two(one) stop bits per character.

cread (-cread)
Enables (disables) the receiver.

clocal (-clocal)
Assumes a line without (with) modem control.

Input Modes

ignbrk (-ignbrk)
Ignores (does not ignore) break on input.

brkint (-brkint)
Signals (does not signal) INTR on break.

ignpar (-ignpar)
Ignores (does not ignore) parity errors.

parmrk (-parmrk)
Marks (does not mark) parity errors (see tty(M)).

inpck (-inpck)
Enables (disables) input parity checking.

istrip (-istrip)
Strips (does not strip) input characters to 7 bits.

inlcr (-inlcr)
Maps (does not map) NL to CR on input.

igncr (-igncr)
Ignores (does not ignore) CR on input.

icrnl (-icrnl)
Maps (does not map) CR to NL on input.

iuclc (-iuclc)
Maps (does not map) uppercase alphabetics to lowercase on input.

ixon (-ixon)
Enables (disables) START/STOP output control. Output is stopped by sending an ASCII DC3 and started by sending an ASCII DC1.

ixany (-ixany)
Allows any character (only DC1) to restart output.

ixoff (-ixoff)
Requests that the system send (not send) START/STOP characters when the input queue is nearly empty/full.

Output Modes

opost (-opost)
Post-processes output (does not post-process output; ignores all other output modes).

olcuc (-olcuc)
Maps (does not map) lowercase alphabetics to uppercase on output.

onlcr (-onlcr)
Maps (does not map) NL to CR-NL on output.

ocrnl (-ocrnl)
Maps (does not map) CR to NL on output.

onocr (-onocr)
Does not (does) output CRs at column zero.

onlret (-onlret)
On the terminal NL performs (does not perform) the CR function.

ofill (-ofill)
Uses fill characters (use timing) for delays.

ofdel (-ofdel)
Fill characters are DELs (NULs).

cr0 cr1 cr2 cr3
Selects style of delay for carriage returns (see tty(M)).

nl1 nl0
Selects style of delay for linefeeds (see tty(M)).

tab0 tab1 tab2 tab3
Selects style of delay for horizontal tabs (see tty(M)).

bs0 bs1
Selects style of delay for backspaces (see tty(M)).

ff0 ff1
Selects style of delay for form feeds (see tty(M)).

vt0 vt1
Selects style of delay for vertical tabs (see tty(M)).

Local Modes

isig (-isig)
Enables (disables) the checking of characters against the special control characters INTR and QUIT.

icanon (-icanon)
Enables (disables) canonical input (ERASE and KILL processing).

xcase (-xcase)
Canonical (unprocessed) upper/lowercase presentation.

echo (-echo)
Echoes back (does not echo back) every character typed.

echoe (-echoe)
Echoes (does not echo) ERASE character as a backspace-space-backspace string. Note: this mode will erase the ERASEed character on many CRT terminals; however, it does not keep track of column position and, as a result, may be confusing on escaped characters, tabs, and backspaces.

echok (-echok)
Echoes (does not echo) NL after KILL character.

lfkc (-lfkc)
The same as echok (-echok); obsolete.

echonl (-echonl)
Echoes (does not echo) NL.

noflsh (-noflsh)
Disables (enables) flush after INTR or QUIT.

Control Assignments

control-character-C
Sets control-character to C, where control-character is erase, kill, intr, quit, eof, min, or time (min and time are used with -icanon; see tty(M)). If C is preceded by a caret (^) (escaped from the shell), then the value used is the corresponding CNTRL character (e.g., "^D" is a CNTRL-D); "^?" is interpreted as DEL and "^-" is interpreted as undefined.

line i
Sets the line discipline to i (0 < i < 127).

Combination Modes

evenp or parity
Enables parenb and cs7.

oddp
Enables parenb, cs7, and parodd.

-parity, -evenp, or -oddp
Disables parenb, and sets cs8.

raw (-raw or cooked)
Enables (disables) raw input and output (no ERASE, KILL, INTR, QUIT, EOT, or output post processing).

nl (-nl)
Unsets (sets) icrnl, onlcr. In addition -nl unsets inlcr, igncr, ocrnl, and onlret.

lcase (-lcase)
Sets (unsets) xcase, iuclc, and olcuc.

LCASE (-LCASE)
Same as lcase (-lcase).

tabs (-tabs or tab3)
Preserves (expands to spaces) tabs when printing.

ek Resets ERASE and KILL characters back to normal CNTRL-H and CNTRL-U.

sane
Resets all modes to some reasonable values. Useful when a terminal's settings have been hopelessly scrambled.

term
Sets all modes suitable for the terminal type term, where term is one of tty33, tty37, vt05, tn300, ti700, or tek.

See Also

tabs(C), ioctl(S), tty(M)

Notes

Many combinations of options make no sense, but no checking is performed.

Name

su - Make the user super user or another user

Syntax

su [userid]

Description

Su allows you to become another user without logging off. The default user name is root (i.e., super-user).

To use su, the appropriate password must be supplied (unless you are already super-user). If the password is correct, su executes a new shell with the user ID set to that of the specified user. To restore normal user ID privileges, type a CNTRL-D to the new shell.

Any additional arguments are passed to the shell, permitting the super-user to run shell procedures with restricted privileges (an arg of the form -c string executes string via the shell). When additional arguments are passed, /bin/sh is always used. When no additional arguments are passed, su uses the shell specified in the password file.

An initial dash (-) causes the environment to be changed to the one that would be expected if the user actually logged in again. This is done by invoking the shell with an arg0 of -su causing the .profile in the home directory of the new user ID to be executed. Otherwise, the environment is passed along with the possible exception of $PATH, which is set to /bin:/etc:/usr/bin for root. Note that .profile can check arg0 for -sh or -su to determine how it was invoked.

Files

/etc/passwd	The system password file
$HOME/.profile	User's profile

See Also

env(C), login(C), sh(C), environ(M)

Name

sync - Update the super block

Syntax

sync

Description

Sync executes the sync system primitive. If the system is to be stopped, sync must be called to insure file system integrity. Shutdown(C) automatically calls sync before shutting down the system.

See Also

sync(S)

Name

tar - Tape or floppy archiver

Syntax

tar [crtux] [bfhlsvw] [tapefile] [blocksize] [tapesize] filel file2...

Description

Tar saves and restores files on magnetic tape, floppy disk, or add-on hard disk. Tar's actions are controlled by a key argument, which contains at least one function letter followed by one or more function modifiers. Other arguments to the command are file or directory names specifying which files are to be dumped or restored. In all cases, a directory name refers to the files and (recursively) subdirectories of that directory.

Tar permits a file to extend across media boundaries.

Specify the function portion of the key by one of the following letters:

c Create a new tape; writing begins on the beginning of the tape instead of after the last file. When you use this command, all previous data is erased.

r The named files are written on the end of the tape.

t The named files are listed each time they occur on the tape. If no file argument is given, all of the names on the tape are listed.

u The named files are added to the tape if either they are not already there or have been modified since last put on the tape.

x The named files are extracted from the tape. If the named file matches a directory whose contents have been written on the tape, this directory is (recursively) extracted. The owner and mode are restored (if possible). If no file argument is given, the entire content of the tape or floppy is extracted. If multiple entries specifying the same file are on the tape, the last version will overwrite all preceding versions.

You can use the following characters in addition to the letter that selects the function desired.

b Causes tar to use the next argument as the blocking factor for tape records. The default is 1, the maximum is 20. This option should only be used with raw magnetic tape archives (see f above). Altos requires a blocking factor of 8 when using the cartridge tape.

f Causes tar to use the next argument as the name of the archive instead of /dev/tar. If the name of the file is '-', tar writes to standard output or reads from standard input, whichever is appropriate. Thus, you can use tar to move hierarchies with the command

```
cd fromdir; tar cf - . | (cd todir; tar xf -)
```

You must use this option with magnetic tape and add-on hard disks. The default is to floppy diskette.

h Archive the contents of the symbolically-linked named files. Tar cv will only archive linkage information; tar chv will archive the contents.

l Tells tar to notify you if it cannot resolve all the links to the files dumped. If this option is not specified, no error messages are printed.

s Obsolete. (Formerly size parameter, used when files didn't cross diskette boundaries.)

v Display the name of each file it treats preceded by the function letter. With the t function, v gives more information about the tape entries than just the name and path.

w Causes tar to display the action to be taken and file name, then wait for user confirmation. If a word or the letter beginning with 'y' is given, the action is performed. Any other input means don't do it.

Files

```
/dev/tar     default input/output device
/tmp/tar*
```

Diagnostics

Complains about bad key characters and tape read/write errors.

Complains if not enough memory is available to hold the link tables.

Tar will tell you to change volumes when the current volume (floppy or tape) becomes full. It expects you to type one or more characters beginning with "y" and then press the Return key.

Notes

This version of tar can read old style tar disks, and the old tar program can read new style tar disks, as long as they do not extend over multiple floppies.

Note that the old version of tar cannot be used to read multiple volume archives created by the new version of tar.

There is no way to ask for the n-th occurrence of a file. Tape errors are handled ungracefully.

The u option can be slow.

The b option should not be used with archives that are going to be updated. If the archive is on a disk file, the b option should not be used at all, as updating an archive stored in this manner can destroy it.

Examples

To dump the directory /usr/john to diskette(s), enter the command

tar cv /usr/john <CR>

To restore the files under /usr/john, check that there is a directory entry for /usr/john. Use the mkdir command to create one, if necessary.

Load the first diskette, and enter

cd /usr/john <CR>
tar xv <CR>

Name

termcap – Terminal capability data base.

Description

The file /etc/termcap is a data base describing terminals. This data base is used by programs such as *vi*(C) and *curses*(S). Terminals are described in *termcap* by giving a set of capabilities and by describing how operations are performed. Padding requirements and initialization sequences are included in *termcap*.

Entries in *termcap* consist of a number of ':' separated fields. The first entry for each terminal gives the names which are known for the terminal, separated by vertical bar (|) characters. The first name is always 2 characters long for compatibility with older systems. The second name given is the most common abbreviation for the terminal, and the last name given should be a long name fully identifying the terminal. The second name should contain no blanks; the last name may well contain blanks for readability.

Capabilities

The following is a list of the capabilities that can be defined for a given terminal. In this list, (P) indicates padding may be specified, (P*) indicates that padding may be based on the number of lines affected, and uppercase names indicate XENIX extensions (except for CC).

Name	Type	Pad?	Description
ae	str	(P)	End alternate character set
al	str	(P*)	Add new blank line
am	bool		Terminal has automatic margins
as	str	(P)	Start alternate character set
bc	str		Backspace if not ^H
BE	str		Bell character
bs	bool		Terminal can backspace with ^H
BS	str		Sent by BACKSPACE key (if not bc)
bt	str	(P)	Back tab
bw	bool		Backspace wraps from column 0 to last column
CC	str		Command character in prototype if terminal settable
cd	str	(P*)	Clear to end of display
ce	str	(P)	Clear to end of line
CF	str		Cursor off
ch	str	(P)	Like cm but horizontal motion only, line stays same
CL	str		Sent by CHAR LEFT key
cl	str	(P*)	Clear screen
cm	str	(P)	Cursor motion

CN	str		Sent by CANCEL key
co	num		Number of columns in a line
CO	str		Cursor on
CR	str		Sent by CHAR RIGHT key
cr	str	(P*)	Carriage return, (default ^M)
cs	str	(P)	Change scrolling region (vt100), like cm
cv	str	(P)	Like ch but vertical only.
CW	str		Sent by CHANGE WINDOW key
da	bool		Display may be retained above
db	bool		Display may be retained below
dB	num		Number of millisec of bs delay needed
dC	num		Number of millisec of cr delay needed
dc	str	(P*)	Delete character
dF	num		Number of millisec of ff delay needed
DK	str		Sent by down arrow key (if not kd)
DL	str		Sent by DELETE key
DL	str		Sent by destructive character delete key
dl	str	(P*)	Delete line
dm	str		Delete mode (enter)
dN	num		Number of millisec of nl delay needed
do	str		Down one line
dT	num		Number of millisec of tab delay needed
ed	str		End delete mode
EE	str		Edit mode end
EG	num		Number of chars taken by ES and EE
ei	str		End insert mode; give `:ei=:` if ic
EN	str		Sent by END key
eo	str		Can erase overstrikes with a blank
ES	str		Edit mode start
ff	str	(P*)	Hardcopy terminal page eject (default ^L)
G1	str		Upper-right (1st quadrant) corner character
G2	str		Upper-left (2nd quadrant) corner character
G3	str		Lower-left (3rd quadrant) corner character
G4	str		Lower-right (4th quadrant) corner character
GD	str		Down-tick character
GE	str		Graphics mode end
GG	num		Number of chars taken by GS and GE
GH	str		Horizontal bar character
GS	str		Graphics mode start
GU	str		Up-tick character
GV	str		Vertical bar character
hc	bool		Hardcopy terminal
hd	str		Half-line down (forward 1/2 linefeed)
HM	str		Sent by HOME key (if not kh)
ho	str		Home cursor (if no cm)
HP	str		Sent by HELP key
hu	str		Half-line up (reverse 1/2 linefeed)
hz	str		Hazeltine; can't print ~'s
ic	str	(P)	Insert character
if	str		Name of file containing is
im	bool		Insert mode (enter); give ':im=:q' if ic
in	bool		Insert mode distinguishes nulls on display

ip	str	(P*)	Insert pad after character inserted
is	str		Terminal initialization string
k0-k9	str		Sent by 'other' function keys 0-9
kb	str		Sent by backspace key
kd	str		Sent by terminal down arrow key
ke	str		Out of 'keypad transmit' mode
KF	str		Key-click off
kh	str		Sent by home key
kl	str		Sent by terminal left arrow key
kn	num		Number of 'other' keys
KO	str		Key-click on
ko	str		Termcap entries for other non-function keys
kr	str		Sent by terminal right arrow key
ks	str		Put terminal in 'keypad transmit' mode
ku	str		Sent by terminal up arrow key
l0-l9	str		Labels on 'other' function keys
LD	str		Sent by line delete key
LF	str		Sent by line feed key
li	num		Number of lines on screen or page
LK	str		Sent by left arrow key (if not kl)
ll	str		Last line, first column (if no cm)
ma	str		Arrow key map, used by vi version 2 only
mi	bool		Safe to move while in insert mode
ml	str		Memory lock on above cursor
MN	str		Sent by minus sign key
MP	str		Multiplan initialization string
MR	str		Multiplan reset string
mu	str		Memory unlock (turn off memory lock)
nc	bool		No correctly working carriage return (DM2500,H2000)
nd	str		Non-destructive space (cursor right)
nl	str	(P*)	Newline character (default \n)
ns	bool		Terminal is a CRT but doesn't scroll
NU	str		Sent by NEXT UNLOCKED CELL key
os	bool		Terminal overstrikes
pc	str		Pad character (rather than null)
PD	str		Sent by PAGE DOWN key
PL	str		Sent by PAGE LEFT key
PR	str		Sent by PAGE RIGHT key
PS	str		Sent by plus sign key
pt	bool		Has hardware tabs (may need to be set with is)
PU	str		Sent by PAGE UP key
RC	str		Sent by RECALC key
RF	str		Sent by TOGGLE REFERENCE key
RK	str		Sent by right arrow key (if not kr)
RT	str		Sent by RETURN key
RT	str		Sent by return key
se	str		End stand out mode
sf	str	(P)	Scroll forwards
sg	num		Number of blank chars left by so or se
so	str		Begin stand out mode
sr	str	(P)	Scroll reverse (backwards)

ta	str	(P)	Tab (other than ^I or with padding)
TB	str		Sent by TAB key
tc	str		Entry of similar terminal - must be last
te	str		String to end programs that use cm
ti	str		String to begin programs that use cm
uc	str		Underscore one char and move past it
ue	str		End underscore mode
ug	num		Number of blank chars left by us or ue
UK	str		Sent by up arrow key (if not ku)
ul	bool		Terminal underlines even though it doesn't overstrike
up	str		Upline (cursor up)
us	str		Start underscore mode
vb	str		Visible bell (may not move cursor)
ve	str		Sequence to end open/visual mode
vs	str		Sequence to start open/visual mode
WL	str		Sent by WORD LEFT key
WR	str		Sent by WORD RIGHT key
xb	bool		Beehive (f1=escape, f2=ctrl C)
xn	bool		A newline is ignored after a wrap (Concept)
xr	bool		Return acts like ce \r \n (Delta Data)
xs	bool		Standard out not erased by writing over it (HP 264?)
xt	bool		Tabs are destructive, magic so char (Teleray 1061)

A Sample Entry

The following entry describes the Concept- 100, and is among the more complex entries in the *termcap* file. (This particular concept entry is outdated, and is used as an example only.)

```
c1 |c100 |concept100:is=\EU\Ef\E7\E5\E8\El\ENH\EK\E\200\Eo&\200:\
      :al=3*\E^R:am:bs:cd=16*\E^C:ce=16\E^S:cl=2*^L:\
       :cm=\Ea%+ %+ :co#80:dc=16\E^A:dl=3*\E^B:ei=\E\200:\
       :eo:im=\E^P:in:ip=16*:li#24:mi:nd=\E=:\
      :se=\Ed\Ee:so=\ED\EE:ta=8\t:ul:up=\E;:vb=\Ek\EK:xn:
```

Entries may continue onto multiple lines by giving a \ as the last character of a line, and that empty fields may be included for readability (here between the last field on a line and the first field on the next). Capabilities in *termcap* are of three types: Boolean capabilities which indicate that the terminal has some particular feature, numeric capabilities giving the size of the terminal or the size of particular delays, and string capabilities, which give a sequence which can be used to perform particular terminal operations.

Types of Capabilities

All capabilities have two letter codes. For instance, the fact that the Concept has 'automatic margins' (i.e. an automatic return and

linefeed when the end of a line is reached) is indicated by the capability am. Hence the description of the Concept includes am. Numeric capabilities are followed by the character '#' and then the value. Thus co which indicates the number of columns the terminal has gives the value '80' for the Concept.

Finally, string valued capabilities, such as ce (clear to end of line sequence) are given by the two character code, an '=', and then a string ending at the next following ':'. A delay in milliseconds may appear after the '=' in such a capability, and padding characters are supplied by the editor after the remainder of the string is sent to provide this delay. The delay can be either a integer, e.g. '20', or an integer followed by an '*', i.e. '3*'. A '*' indicates that the padding required is proportional to the number of lines affected by the operation, and the amount given is the per-affected-unit padding required. When a '*' is specified, it is sometimes useful to give a delay of the form '3.5' specify a delay per unit to tenths of milliseconds.

A number of escape sequences are provided in the string valued capabilities for easy encoding of characters there. A \E maps to an ESCAPE character, ^x maps to a control-x for any appropriate x, and the sequences \n \r \t \b \f give a newline, return, tab, backspace and formfeed. Finally, characters may be given as three octal digits after a \, and the characters ^ and \ may be given as \^ and \\. If it is necessary to place a : in a capability it must be escaped in octal as \072. If it is necessary to place a null character in a string capability it must be encoded as \200. The routines that deal with *termcap* use C strings, and strip the high bits of the output very late so that a \200 comes out as a \000 would.

Preparing Descriptions

We now outline how to prepare descriptions of terminals. The most effective way to prepare a terminal description is by imitating the description of a similar terminal in *termcap* and to build up a description gradually, using partial descriptions with *ex* to check that they are correct. Be aware that a very unusual terminal may expose deficiencies in the ability of the *termcap* file to describe it or bugs in *ex*. To easily test a new terminal description you can set the environment variable TERMCAP to a pathname of a file containing the description you are working on and the editor will look there rather than in */etc/termcap*. TERMCAP can also be set to the termcap entry itself to avoid reading the file when starting up the editor.

Basic capabilities

The number of columns on each line for the terminal is given by the co numeric capability. If the terminal is a CRT, then the number of lines on the screen is given by the li capability. If the terminal wraps around to the beginning of the next line when it reaches the right margin, then it should have the am capability. If the terminal can clear its screen, then this is given by the cl string capability. If the terminal can backspace, then it should have the bs capability,

unless a backspace is accomplished by a character other than ^H in which case you should give this character as the **bc** string capability. If it overstrikes (rather than clearing a position when a character is struck over) then it should have the **os** capability.

A very important point here is that the local cursor motions encoded in *termcap* are undefined at the left and top edges of a CRT terminal. The editor will never attempt to backspace around the left edge, nor will it attempt to go up locally off the top. The editor assumes that feeding off the bottom of the screen will cause the screen to scroll up, and the **am** capability tells whether the cursor sticks at the right edge of the screen. If the terminal has switch selectable automatic margins, the *termcap* file usually assumes that this is on, i.e. **am**.

These capabilities suffice to describe hardcopy and 'glass-tty' terminals. Thus the model 33 teletype is described as

t3|33|tty33:co#72:os

while the Lear Siegler ADM-3 is described as

cl|adm3|si adm3:am:bs:cl=^Z:li#24:co#80

Cursor addressing

Cursor addressing in the terminal is described by a **cm** string capability, with *printf*(S) like escapes %x in it. These substitute to encodings of the current line or column position, while other characters are passed through unchanged. If the **cm** string is thought of as being a function, then its arguments are the line and then the column to which motion is desired, and the % encodings have the following meanings:

%d	as in *printf*, 0 origin
%2	like %2d
%3	like %3d
%.	like %c
%+ x	adds *x* to value, then %.
%>xy	if value > x adds y, no output.
%r	reverses order of line and column, no output
%i	increments line/column (for 1 origin)
%%	gives a single %
%n	exclusive or row and column with 0140 (DM2500)
%B	BCD (16*(x/10)) + (x%10), no output.
%D	Reverse coding (x-2*(x%16)), no output. (Delta Data).

Consider the HP2645, which, to get to row 3 and column 12, needs to be sent \E&a12c03Y padded for 6 milliseconds. Note that the order of the rows and columns is inverted here, and that the row and column are printed as two digits. Thus its **cm** capability is 'cm=6\E&%r%2c%2Y'. The Microterm ACT-IV needs the current row and column sent preceded by a ^T, with the row and column simply encoded in binary, 'cm=^T%.%.'. Terminals which use '%.'

need to be able to backspace the cursor (bs or bc), and to move the cursor up one line on the screen (up introduced below). This is necessary because it is not always safe to transmit \t, \n ^D and \r, as the system may change or discard them.

A final example is the LSI ADM-3a, which uses row and column offset by a blank character, thus 'cm=\E=%+ %+ '.

Cursor motions

If the terminal can move the cursor one position to the right, leaving the character at the current position unchanged, then this sequence should be given as nd (non-destructive space). If it can move the cursor up a line on the screen in the same column, this should be given as up. If the terminal has no cursor addressing capability, but can home the cursor (to very upper left corner of screen) then this can be given as ho; similarly a fast way of getting to the lower left hand corner can be given as ll; this may involve going up with up from the home position, but the editor will never do this itself (unless ll does) because it makes no assumption about the effect of moving up from the home position.

Area clears

If the terminal can clear from the current position to the end of the line, leaving the cursor where it is, this should be given as ce. If the terminal can clear from the current position to the end of the display, then this should be given as cd. The editor only uses cd from the first column of a line.

Insert/delete line

If the terminal can open a new blank line before the line where the cursor is, this should be given as al; this is done only from the first position of a line. The cursor must then appear on the newly blank line. If the terminal can delete the line which the cursor is on, then this should be given as dl; this is done only from the first position on the line to be deleted. If the terminal can scroll the screen backwards, then this can be given as sb, but just al suffices. If the terminal can retain display memory above then the da capability should be given; if display memory can be retained below then db should be given. These let the editor understand that deleting a line on the screen may bring non-blank lines up from below or that scrolling back with sb may bring down non-blank lines.

Insert/delete character

There are two basic kinds of intelligent terminals with respect to insert/delete character which can be described using *termcap*. The most common insert/delete character operations affect only the characters on the current line and shift characters off the end of the line. Other terminals, such as the Concept 100 and the Perkin Elmer Owl, make a distinction between typed and untyped blanks on the screen,

shifting upon an insert or delete only to an untyped blank on the screen which is either eliminated, or expanded to two untyped blanks. You can find out which kind of terminal you have by clearing the screen and then typing text separated by cursor motions. Type 'abc def' using local cursor motions (not spaces) between the 'abc' and the 'def'. Then position the cursor before the 'abc' and put the terminal in insert mode. If typing characters causes the rest of the line to shift rigidly and characters to fall off the end, then your terminal does not distinguish between blanks and untyped positions. If the 'abc' shifts over to the 'def' which then move together around the end of the current line and onto the next as you insert, you have the second type of terminal, and should give the capability in, which stands for 'insert null'. If your terminal does something different and unusual then you may have to modify the editor to get it to use the insert mode your terminal defines. No known terminals have an insert mode not falling into one of these two classes.

The editor can handle both terminals that have an insert mode and terminals which send a simple sequence to open a blank position on the current line. Give as im the sequence to get into insert mode, or give it an empty value if your terminal uses a sequence to insert a blank position. Give as ei the sequence to leave insert mode (give this, with an empty value also if you gave im an empty value). Now give as ic any sequence needed to be sent just before sending the character to be inserted. Most terminals with a true insert mode will not give ic, terminals which send a sequence to open a screen position should give it here. (Insert mode is preferable to the sequence to open a position on the screen if your terminal has both.) If post insert padding is needed, give this as a number of milliseconds in ip (a string option). Any other sequence which may need to be sent after an insert of a single character may also be given in ip.

It is occasionally necessary to move around while in insert mode to delete characters on the same line (e.g. if there is a tab after the insertion position). If your terminal allows motion while in insert mode you can give the capability mi to speed up inserting in this case. Omitting mi will affect only speed. Some terminals (notably Datamedia's) must not have mi because of the way their insert mode works.

Finally, you can specify delete mode by giving dm and ed to enter and exit delete mode, and dc to delete a single character while in delete mode.

Highlighting, underlining, and visible bells

If your terminal has sequences to enter and exit standout mode these can be given as so and se respectively. If there are several flavors of standout mode (such as inverse video, blinking, or underlining - half bright is not usually an acceptable 'standout' mode unless the terminal is in inverse video mode constantly) the preferred mode is inverse video by itself. If the code to change into or out of standout mode leaves one or even two blank spaces on the

screen, as the TVI 912 and Teleray 1061 do, this is acceptable, and although it may confuse some programs slightly, it can't be helped.

Codes to begin underlining and end underlining can be given as us and ue respectively. If the terminal has a code to underline the current character and move the cursor one space to the right, such as the Microterm Mime, this can be given as uc. (If the underline code does not move the cursor to the right, give the code followed by a nondestructive space.)

If the terminal has a way of flashing the screen to indicate an error quietly (a bell replacement) then this can be given as vb; it must not move the cursor. If the terminal should be placed in a different mode during open and visual modes of *ex*, this can be given as vs and ve, sent at the start and end of these modes respectively. These can be used to change, e.g., from a underline to a block cursor and back.

If the terminal needs to be in a special mode when running a program that addresses the cursor, the codes to enter and exit this mode can be given as ti and te. This arises, for example, from terminals like the Concept with more than one page of memory. If the terminal has only memory relative cursor addressing and not screen relative cursor addressing, a one screen-sized window must be fixed into the terminal for cursor addressing to work properly.

If your terminal correctly generates underlined characters (with no special codes needed) even though it does not overstrike, then you should give the capability ul. If overstrikes are erasable with a blank, then this should be indicated by giving eo.

Keypad

If the terminal has a keypad that transmits codes when the keys are pressed, this information can be given. Note that it is not possible to handle terminals where the keypad only works in local (this applies, for example, to the unshifted HP 2621 keys). If the keypad can be set to transmit or not transmit, give these codes as ks and ke. Otherwise the keypad is assumed to always transmit. The codes sent by the left arrow, right arrow, up arrow, down arrow, and home keys can be given as kl, kr, ku, kd, and kh respectively. If there are function keys such as f0, f1, ..., f9, the codes they send can be given as k0, k1, ..., k9. If these keys have labels other than the default f0 through f9, the labels can be given as l0, l1, ..., l9. If there are other keys that transmit the same code as the terminal expects for the corresponding function, such as clear screen, the *termcap* 2 letter codes can be given in the ko capability, for example, ':ko=cl,ll,sf,sb:', which says that the terminal has clear, home down, scroll down, and scroll up keys that transmit the same thing as the cl, ll, sf, and sb entries.

The ma entry is also used to indicate arrow keys on terminals which have single character arrow keys. It is obsolete but still in use in

version 2 of vi, which must be run on some minicomputers due to memory limitations. This field is redundant with **kl**, **kr**, **ku**, **kd**, and **kh**. It consists of groups of two characters. In each group, the first character is what an arrow key sends, the second character is the corresponding vi command. These commands are **h** for **kl**, **j** for **kd**, **k** for **ku**, **l** for **kr**, and **H** for **kh**. For example, the mime would be :ma=^Kj^Zk^Xl: indicating arrow keys left (^H), down (^K), up (^Z), and right (^X). (There is no home key on the mime.)

Miscellaneous

If the terminal requires other than a null (zero) character as a pad, then this can be given as **pc**.

If tabs on the terminal require padding, or if the terminal uses a character other than ^I to tab, then this can be given as **ta**.

Hazeltine terminals, which don't allow '~' characters to be printed should indicate **hz**. Datamedia terminals, which echo carriage-return linefeed for carriage return and then ignore a following linefeed should indicate **nc**. Early Concept terminals, which ignore a linefeed immediately after an **am** wrap, should indicate **xn**. If an erase-eol is required to get rid of standout (instead of merely writing on top of it), **xs** should be given. Teleray terminals, where tabs turn all characters moved over to blanks, should indicate **xt**. Other specific terminal problems may be corrected by adding more capabilities of the form **x***x*.

Other capabilities include **is**, an initialization string for the terminal, and **if**, the name of a file containing long initialization strings. These strings are expected to properly clear and then set the tabs on the terminal, if the terminal has settable tabs. If both are given, **is** will be printed before **if**. This is useful where **if** is */usr/lib/tabset/std* but **is** clears the tabs first.

Similar Terminals

If there are two very similar terminals, one can be defined as being just like the other with certain exceptions. The string capability **tc** can be given with the name of the similar terminal. This capability must be *last* and the combined length of the two entries must not exceed 1024. Since *termlib* routines search the entry from left to right, and since the **tc** capability is replaced by the corresponding entry, the capabilities given at the left override the ones in the similar terminal. A capability can be cancelled with **xx@** where xx is the capability. For example:

hn|2621nl:ks@:ke@:tc=2621:

This defines a 2621nl that does not have the **ks** or **ke** capabilities, and hence does not turn on the function key labels when in visual mode. This is useful for different modes for a terminal, or for different user preferences.

Files

/etc/termcap File containing terminal descriptions

See Also

ex(C), curses(S), termcap(S), tset(C), vi(C), more(C)

Credit

This utility was developed at the University of California at Berkeley and is used with permission.

Notes

Ex allows only 256 characters for string capabilities, and the routines in *termcap*(S) do not check for overflow of this buffer. The total length of a single entry (excluding only escaped newlines) may not exceed 1024.

The ma, vs, and ve entries are specific to the *vi* program.

Not all programs support all entries. There are entries that are not supported by any program.

Name

test - Tests conditions

Syntax

test expr
[expr]

Description

test evaluates the expression expr, and if its value is true then returns zero exit status; otherwise, a non zero exit status is returned. test returns a non zero exit if there are no arguments.

The following primitives are used to construct expr.

-r file True if the file exists and is readable.

-w file True if the file exists and is writable.

-x file True if the file exits and is executable.

-f file True if the file exists and is not a directory.

-d file True if the file exists and is a directory.

-c file True if file exists and is a character special file.

-b file True if file exists and is a block special file.

-u file True if file exists and its set-user-ID bit is set.

-g file True if file exists and its set-group-ID bit is set.

-k file True if file exists and its sticky bit is set.

-s file True if the file exists and has a size greater than zero.

-t [fildes]
True if the open file whose file descriptor number is fildes (1 by default) is associated with a terminal device.

-z sl True if the length of string sl is zero.

-n s1 True if the length of the string s1 is nonzero.

s1 = s2 True if the strings s1 and s2 are equal.

s1 != s2 True if the strings s1 and s2 are not equal.

s1 True if s1 is not the null string.

n1 -eq n2
True if the integers n1 and n2 are algebraically equal. Any of the comparisons -ne, -gt, -ge, -lt, or -le may be used in place of -eq.

These primaries may be combined with the following operators:

! unary negation operator

-a binary and operator

-o binary or operator

(expr) parentheses for grouping.

Notice that all the operators and flags are separate arguments to test. Notice also that parentheses are meaningful to the Shell and must be escaped.

See Also

sh(C), find(C)

Warning

In the second form of the command (i.e., the one that uses [], rather than the word test), the square brackets must be delimited by blanks.

Name

transp - Set up transparent printer

Syntax

/etc/transp /dev/ttyn /dev/lpm

Description

The transp command is used to set up the proper device files for serial printers hooked to the back of an Altos II terminal. The /dev/ttyn argument specifies which tty device (n) has the daisy-chained printer attached to its auxiliary port. This terminal must be an Altos II terminal. The /dev/lpm argument specifies the name of the printer device (e.g., /dev/lp2). If the specified printer is /dev/lp, then this will be the default printer in the system.

The transp command creates a link between a tty port and a printer port. Once this link is established, it is not necessary to perform this command again unless the printer is moved. Output destined for a printer hooked to the back of an Altos II switches the terminal into transparent mode, and the terminal becomes inoperative while the printer is working. When printing stops (i.e., the user program closes the print device), the terminal becomes active again.

See Also

mknod(C), tty(M)

Examples

transp /dev/tty6 /dev/lp

This command sets up serial port 6, which has an Altos II terminal attached, to also be the default printer port in the system.

```
transp /dev/tty5 /dev/lp2
transp tty5 lp2
transp 5 2
```

All these commands are equivalent. They set up serial port 5 to also be line printer 2 in the system.

Name

true - Returns with a zero exit value

Syntax

true

Description

True does nothing except return with a zero exit value. False(C), true's counterpart, does nothing except return a nonzero exit value. True is typically used in shell procedures such as:

```
while true
do
     command
done
```

See Also

sh(C), False(C)

Diagnostics

True has exit status zero.

Name

tset - Sets terminal modes.

Syntax

tset [-] [-hrsuIQS] [-e[c] [-E[c] [-k[c]]
[-m [ident][test baudrate]:type] [type]

Description

Tset causes terminal dependent processing such as setting erase and kill characters, setting or resetting delays, and the like. It is driven by the /etc/ttytype and /etc/termcap files.

The type of terminal is specified by the type argument. The type may be any type given in /etc/termcap. If type is not specified, the terminal type is the value of the environment variable TERM, unless the -h flag is set or any -m argument is given. In this case the type is read from /etc/ttytype (the port name to terminal type database). The port name is determined by a ttyname(S) call on the diagnostic output. If the port is not found in /etc/ttytype the terminal type is set to unknown.

Ports for which the terminal type is indeterminate are identified in /etc/ttytype as dialup, plugboard, etc. The user can specify how these identifiers should map to an actual terminal type. The mapping flag, -m, is followed by the appropriate identifier (a four-character or longer substring is adequate), an optional test for baud rate, and the terminal type to be used if the mapping conditions are satisfied. If more than one mapping is specified, the first correct mapping prevails. A missing identifier matches all identifiers. Baud rates are specified as with stty(C), and are compared with the speed of the diagnostic output. The test may be any combination of: >, =, <, @, and !. (Note: @ is a synonym for = and ! inverts the sense of the test. Remember to escape characters meaningful to the shell.)

If the type as determined above begins with a question mark, the user is asked if he really wants that type. A null response means to use that type; otherwise, another type can be entered which will be used instead. (The question mark must be escaped to prevent filename expansion by the shell.) The -e flag sets the erase character to be the named character c on all terminals, so to override this option one can say -e#. The default for c is the backspace character on the terminal, usually CNTRL-H. The -E flag is identical to -e except that it only operates on terminals that can

backspace. The -k option works similarly, with c defaulting to CNTRL-U. No kill processing is done if -k is not specified. In all of these flags, "^X" where X is any character is equivalent to CNTRL-X.

The - option prints the terminal type on the standard output; this can be used to get the terminal type by saying:

```
set termtype = `tset-`
```

If no other options are given, tset operates in "fast mode" and only outputs the terminal type, bypassing all other processing. The -s option outputs "setenv" commands (if your default shell is csh) or "export" and assignment commands (if your default shell is the Bourne shell); the -S option only outputs the strings to be placed in the environment variables. For the -s option with the Bourne shell, use:

```
`tset -s ...`


tset -s ... > /tmp/tset$$
/tmp/tset$$
rm /tmp/tset$$
```

If you are using csh, use:

```
set noglob
set term=('tset -S ....')
setenv TERM $term[1]
setenv TERMCAP "$term[2]"
unset term
unset noglob
```

The -r option prints the terminal type on the diagnostic output. The -Q option suppresses printing the "Erase set to" and "Kill set to" messages. The -I option suppresses outputting the terminal initialization strings.

Tset is most useful when included in the .login (for csh(C)) or .profile (for sh(C)) file executed automatically at login, with -m mapping used to specify the terminal type you most frequently dial in on.

Examples

```
tset gt42

tset -mdialup\>300:adm3a -mdialup:dw2 -Qr -e#

tset -m dial:ti733 -m plug:\?hp2621 -m unknown:\? -e -k ^U
```

Files

/etc/ttytype	Port name to terminal type map database
/etc/termcap	Terminal capability database

See Also

setenv(C), ttys(F), termcap(F), stty(C)

Credit

This utility was developed at the University of California at Berkeley and is used with permission.

Name

tty - Get terminal name

Syntax

tty [-s]

Description

Tty prints the pathname of the user's terminal on the standard output. The -s option inhibits printing, allowing you to test just the exit code.

Exit Codes

Ø if the standard input is a terminal, 1 otherwise.

Diagnostics

'not a tty' if the standard input file is not a terminal.

Name

tty – General terminal interface.

Description

This section describes both a particular special file and the general nature of the terminal interface.

The file /dev/tty is, in each process, a synonym for the control terminal associated with the process group of that process, if any. It is useful for programs or shell sequences that wish to be sure of writing messages on the terminal no matter how output has been redirected. It can also be used for programs that demand the name of a file for output, when typed output is desired and it is tiresome to find out what terminal is currently in use.

All asynchronous communications ports use the same general interface, no matter what hardware is involved. The remainder of this section discusses the common features of this interface.

When a terminal file is opened, it normally causes the process to wait until a connection is established. In practice, users' programs seldom open these files; they are opened by *getty*(M) and become a user's standard input, output, and error files. The very first terminal file opened by the process group leader of a terminal file 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 below. The control terminal is inherited by a child process during a *fork*(S). A process can break this association by changing its process group using *setpgrp*(S).

A terminal associated with one of these files ordinarily operates in full-duplex mode. Characters may be typed at any time, even while output is occurring, and are only lost when the system's character input buffers become completely full, which is rare, or when the user has accumulated the maximum allowed number of input characters that have not yet been read by some program. Currently, this limit is 256 characters. When the input limit is reached, 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 attempting to read will be suspended until an entire line has been typed. Also, no matter how many characters are requested in the read call, at most one line will be returned. It is not, however, necessary to read a whole line at once; any number of characters may be requested in a read, even one, without losing information.

Erase and kill processing is normally done during input. By default, a CNTRL-H or BACKSPACE erases the last character typed, except

that it will not erase beyond the beginning of the line. By default, a CNTRL-U kills (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. Both the erase and kill characters may be entered literally by preceding them with the escape character (\). In this case the escape character is not read. The erase and kill characters may be changed (see *stty*(C)).

Certain characters have special functions on input. These functions and their default character values are summarized as follows:

INTR (Rubout or ASCII DEL) Generates an *interrupt* signal which is sent to all processes with the associated control terminal. Normally, each such process is forced to terminate, but arrangements may be made either to ignore the signal or to receive a trap to an agreed-upon location; see *signal*(S).

QUIT (CNTRL-\ or ASCII FS) Generates a *quit* signal. Its treatment is identical to the interrupt signal except that, unless a receiving process has made other arrangements, it will not only be terminated but a core image file (called core) will be created in the current working directory.

ERASE (CNTRL-H) Erases the preceding character. It will not erase beyond the start of a line, as delimited by a NL, EOF, or EOL character.

KILL (CNTRL-U) Deletes the entire line, as delimited by a NL, EOF, or EOL character.

EOF (CNTRL-D or ASCII EOT) May be used to generate an end-of-file from a terminal. When received, all the characters waiting to be read are immediately passed to the program, without waiting for a newline, and the EOF is discarded. Thus, if there are no characters waiting, which is to say the EOF occurred at the beginning of a line, zero characters will be passed back, which is the standard end-of-file indication.

NL (ASCII LF) Is the normal line delimiter. It cannot be changed or escaped.

EOL (ASCII NUL) Is an additional line delimiter, like NL. It is not normally used.

STOP (CNTRL-S or ASCII DC3) Can be used to temporarily suspend output. It is useful with CRT terminals to prevent output from disappearing before it can be read. While output is suspended, STOP characters are ignored and not read.

START (CNTRL-Q or ASCII DC1) Is used to resume output which has been suspended by a STOP character. While output is not suspended, START characters are ignored and not read. The start/stop characters cannot be changed or escaped.

The character values for INTR, QUIT, ERASE, KILL, EOF, and EOL may be changed to suit individual tastes. The ERASE, KILL, and EOF characters may be escaped by a preceding \ character, in which case no special function is carried out.

When the carrier signal from the dataset drops, a *hangup* signal is sent to all processes that have this terminal as the control terminal. Unless other arrangements have been made, this signal causes the processes to terminate. If the hangup signal is ignored, any subsequent read returns with an end-of-file indication. Thus programs that read a terminal and test for end-of-file can terminate appropriately when hung up on.

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 a given limit. When the queue has drained down to the given threshold, the program is resumed.

Several *ioctl*(S) system calls apply to terminal files. The primary calls use the following structure, defined in the file **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   DEL
1   VQUIT   FS
2   VERASE  BKSP
3   VKILL   CNTRL-U, CNTRL-H,
4   VEOF    EOT
5   VEOL    NUL
6   Reserved
7   Reserved
```

The *c_iflag* field describes the basic terminal input control:

IGNBRK	0000001	Ignores break condition
BRKINT	0000002	Signals interrupt on break
IGNPAR	0000004	Ignores characters with parity errors
PARMRK	0000010	Marks parity errors
INPCK	0000020	Enables input parity check
ISTRIP	0000040	Strips character
INLCR	0000100	Maps NL to CR on input
IGNCR	0000200	Ignores CR
ICRNL	0000400	Maps CR to NL on input
IUCLC	0001000	Maps uppercase to lowercase on input
IXON	0002000	Enables start/stop output control
IXANY	0004000	Enables any character to restart output
IXOFF	0010000	Enables start/stop input control

If IGNBRK is set, the break condition (a character framing error with data all zeros) is ignored, that is, not put on the input queue and therefore not read by any process. Otherwise, if BRKINT is set the break condition will generate an interrupt signal and flush both the input and output queues. If IGNPAR is set, characters with other framing and parity errors are ignored.

If PARMRK is set, a character with a framing or parity error which is not ignored is read as the 3-character sequence: 0377, 0, X, where X is the data of the character received in error. To avoid ambiguity in this case, if ISTRIP is not set, a valid character of 0377 is read as 0377, 0377. If PARMRK is not set, a framing or parity error which is not ignored is read as the character NUL (0).

If INPCK is set, input parity checking is enabled. If INPCK is not set, input parity checking is disabled. This allows output parity generation without input parity errors.

If ISTRIP is set, valid input characters are first stripped to 7-bits, otherwise all 8-bits are processed.

If INLCR is set, a received NL character is translated into a CR character. If IGNCR is set, a received CR character is ignored (not read). Otherwise if ICRNL is set, a received CR character is translated into a NL character.

If IUCLC is set, a received 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 characters when the input queue is nearly empty and STOP characters when nearly full.

The initial input control value is all bits clear.

The *c_oflag* field specifies the system treatment of output:

OPOST	0000001	Postprocesses output
OLCUC	0000002	Maps lowercase to uppercase on output
ONLCR	0000004	Maps NL to CR-NL on output
OCRNL	0000010	Maps CR to NL on output
ONOCR	0000020	No CR output at column 0
ONLRET	0000040	NL performs CR function
OFILL	0000100	Uses fill characters for delay
OFDEL	0000200	Fills is DEL, else NUL
NLDLY	0000400	Selects newline delays:
NL0	0	
NL1	0000400	
CRDLY	0003000	Selects carriage return delays:
CR0	0	
CR1	0001000	
CR2	0002000	
CR3	0003000	
TABDLY	0014000	Selects horizontal tab delays:
TAB0	0	
TAB1	0004000	
TAB2	0010000	
TAB3	0014000	Expands tabs to spaces
BSDLY	0020000	Selects backspace delays:
BS0	0	
BS1	0020000	
VTDLY	0040000	Selects vertical tab delays:
VT0	0	
VT1	0040000	
FFDLY	0100000	Selects form feed delays:
FF0	0	
FF1	0100000	

If OPOST is set, output characters are post-processed as indicated by the remaining flags, otherwise characters are transmitted without change.

If OLCUC is set, a 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 perform the carriage return function; the column pointer will be set to 0 and the delays specified for CR will be used. Otherwise the NL character is assumed to perform the linefeed function; the column pointer will remain unchanged. The column pointer is also set to 0 if the CR character is actually transmitted.

The delay bits specify how long transmission stops to allow for mechanical or other movement when certain characters are sent to the terminal. In all cases a value of 0 indicates no delay. If OFILL is set, fill characters will be transmitted for delay instead of a timed delay. This is useful for high baud rate terminals which need only a minimal delay. If OFDEL is set, the fill character is DEL, otherwise NUL.

If a form feed or vertical tab delay is specified, it lasts for about 2 seconds.

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, 2 fill characters will be transmitted.

Carriage return delay type 1 is dependent on the current column position, type 2 is about 0.10 seconds, and type 3 is about 0.15 seconds. If OFILL is set, delay type 1 transmits 2 fill characters, and type 2 transmits 4 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, 2 fill characters will be transmitted for any delay.

Backspace delay lasts about 0.05 seconds. If OFILL is set, 1 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
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	Sends two stop bits, else one
CREAD	0000200	Enables receiver
PARENB	0000400	Parity enable
PARODD	0001000	Odd parity, else even
HUPCL	0002000	Hangs up on last close
CLOCAL	0004000	Local line, else dial-up

The CBAUD bits specify the baud rate. The zero baud rate, B0, is used to hang up the connection. If B0 is specified, the data-terminal-ready signal will not be asserted. Without this signal, the line is disconnected if connected through a modem. 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, 2 stop bits are used, otherwise 1 stop bit. For example, at 110 baud, 2 stops bits are required.

If PARENB is set, parity generation and detection is enabled and a parity bit is added to each character. If parity is enabled, the PARODD flag specifies odd parity if set, otherwise even parity is used.

If CREAD is set, the receiver is enabled. Otherwise no characters will be received.

If HUPCL is set, the line will be disconnected when the last process with the line open closes it or terminates. That is, the data-terminal-ready signal will not be asserted.

If CLOCAL is set, the line is assumed to be a local, direct connection with no modem control. The data-terminal-ready and request-to-send signals are asserted, but incoming modem signals are ignored. If CLOCAL is not set, modem control is assumed. This means the data-terminal-ready and request-to-send signals are asserted. Also,

the carrier-detect signal must be returned before communications can proceed.

The initial hardware control value after open is B9600, 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	Enables echo
ECHOE	0000020	Echoes erase character as BS-SP-BS
ECHOK	0000040	Echoes NL after kill character
ECHONL	0000100	Echoes NL
NOFLSH	0000200	Disables flush after interrupt or quit
XCLUDE	0100000	Exclusive use of the line.

If ISIG is set, each input character is checked against the special control characters INTR 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 (e.g. 0377).

If ICANON is set, canonical processing is enabled. This enables the erase and kill edit functions, and the assembly of input characters into lines 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 VMIN characters have been received or the timeout value VTIME has expired. This allows fast bursts of input to be read efficiently while still allowing single character input. The VMIN and VTIME 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 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 will clear the last character from a CRT screen. If ECHOE is set and ECHO is not set, the erase character is echoed as ASCII SP BS. If ECHOK is set, the NL character will be echoed after the kill character to emphasize that the line will be deleted. Note that an escape character preceding the erase or kill character removes any special function. If ECHONL is set, the NL character will be echoed even if ECHO is not set. This is useful for terminals set to local echo (so-called half duplex). Unless escaped, the EOF character is not echoed. Because EOT is the default EOF character, this prevents terminals that respond to EOT from hanging up.

If NOFLSH is set, the normal flush of the input and output queues associated with the quit and interrupt characters will not be done.

If XCLUDE is set, any subsequent attempt to open the tty device using *open*(S) will fail for all users except the super-user. If the call fails, it returns EBUSY in *errno*. XCLUDE is useful for programs which must have exclusive use of a communications line. It is not intended for the line to the program's controlling terminal. XCLUDE must be cleared before the setting program terminates, otherwise subsequent attempts to open the device will fail.

The initial line-discipline control value is all bits clear.

The primary *ioctl*(S) system calls have the form:

```
ioctl (fildes, command, arg)
struct termio *arg;
```

The commands using this form are:

TCGETA — Gets the parameters associated with the terminal and stores them in the *termio* structure referenced by **arg**.

TCSETA — Sets the parameters associated with the terminal from the structure referenced by **arg**. The change is immediate.

TCSETAW — Waits for the output to drain before setting the new parameters. This form should be used when changing parameters that will affect output.

TCSETAF — Waits for the output to drain, then flushes the input queue and sets the new parameters.

Additional *ioctl*(S) calls have the form:

```
ioctl (fildes, command, arg)
int arg;
```

The commands using this form are:

TCSBRK	Waits for the output to drain. If *arg* is 0, then sends a break (zero bits for 0.25 seconds).
TCXONC	Starts/stops control. If *arg* is 0, suspends output; if 1, restarts suspended output.
TCFLSH	If *arg* is 0, flushes the input queue; if 1, flushes the output queue; if 2, flushes both the input and output queues.

Files

/dev/tty

/dev/tty*

/dev/console

See Also

stty(C), ioctl(S)

Name

ttys - Login terminals file.

Description

The /etc/ttys file contains a list of the device special files associated with possible login terminals, and defines which files are to be opened by the *init*(M) program on system start-up.

The file contains one or more entries of the form

state mode name

The *name* must be the filename of a device special file. Only the filename may be supplied, the path is assumed to be /dev. If *state* is "1", the file is enabled for logins; if "0", the file is disabled. The *mode* is used as an argument to the *getty*(M) program. It defines the line speed and type of device associated with the terminal. A list of arguments is provided in *getty*(M).

For example, the entry "16tty02" means the serial line tty02 is to be opened for logging in at 9600 baud.

Files

/etc/ttys

See Also

init(M), getty(M), enable(C), disable(C)

Notes

The /etc/ttys file may only be edited when the system is in system maintenance mode. See the XENIX *Operations Guide*.

Name

txset - Set transparent printer escape sequence

Syntax

/etc/txset /dev/ttyN

Description

The txset command is used to supply a terminal escape sequence to the operating system for use when redirecting output to a printer attached to a Altos II terminal. There are two such sequences, one for requesting redirection to the printer, and one for reversing the process. The /dev/ttyn argument specifies which tty device (N) has an Altos II with printer attached to its auxiliary port.

Two ioctl calls have been added to the serial driver to accommodate the setting of these escape sequences. They are:

```
#define   TIOXESCO  (('t'<<8) |21)
#define   TIOXESCC  (('t'<<8) |22)
```

The TIOXESCO call supplies the redirection sequence, TIOX-ESCC supplies the reverse. The escape sequences and a sample invocation of the ioctls are shown below.

```
char oesc[] = {27,'[','=','1','1',27,'[','5','i',' '};
char cesc[] = {27,'[','4','i',' '};

ioctl(fd,TIOXESCO,oesc);
ioctl(fd,TIOXESCC,cesc);
```

The "fd" field is an int, resulting from the invocation of an open(S) system call. The final argument to either ioctl is the address of an escape sequence, null terminated, in user data space. Txset(C) is used by the transp(C) script as a consequence of initializing a transparent printer.

This mechanism may be used to supply escape sequences for terminals other than Altos II. A program similar to txset may be easily written to generate the appropriate ioctl calls.

See Also

transp(C), mknod(C), tty(M), ioctl(S)

Examples

```
txset /dev/tty6
```

This command sets up serial port 6 for an Altos II with a printer attached.

Name

ua -- User administration

Syntax

ua [-h]

Description

Ua is used for the addition, deletion and modification of users and groups. It provides an effective means for maintaining the system password (/etc/passwd) and system group (/etc/group) files.

The command is implemented using the termcap and curses facilities from UC Berkeley. It must be run interactively from a terminal which is defined within /etc/termcap.

This command should only be run by Super User -- improper results may occur if it is run by a normal user.

The following option is interpreted by ua:

-h Displays the program's current version and copyright notice as well as a short description of the program's functions.

Ua displays its legal commands at the top of the screen. The "Command?" prompt at the bottom of the screen indicates that ua is awaiting input. The command language syntax is:

[add|delete|show|change] [user <name> | group <name>]

show [Users | Groups]

[help | ?]

! [<shell command(s)>]

quit

The system responds as soon as the first letter of a command is typed. Full command words are not acceptable as input. The case of each word is significant: "group" is not the same as "Group."

Typing an interrupt (usually RUBOUT or DEL) will cause ua to immediately return to the top-level command interpreter.

Add allows the addition of a new user or group. After user/group is specified and a new name provided, the system immediately enters the change command so as to allow modification of the new entry. At the conclusion of the change command the addition is made. If a directory already exists for a new user, it is not removed. All files under /etc/newuser are copied to the new directory during the user installation process. Typically /etc/newuser will contain the standard versions of the following files: .cshrc, .login, .logout, .profile. The initial value given to a new user ID is one more than the maximum user ID currently in use. The same is true for a new group ID.

Delete allows the deletion of an existing user or group. Deleting a user optionally also deletes his directory and all files contained within it. Deleting a user will not cause all files throughout the system owned by the user to be deleted -- only those beneath his directory. Thus, some files may have an "unknown" owner after a user is deleted. And, if a user is later added with the same user ID as the deleted user, these files will suddenly belong to the new user. The same problem may arise with the deletion and later addition of a group.

Show will show an individual user or group or all users or groups. The word "show" may be omitted if desired.

Change allows the modification of any existing user or group. A special display mode is entered with a menu of fields for selection of the item to be modified. Typing RETURN or LINE FEED in response to a field change request will empty the field. Changes to a user or group change the corresponding entries in the /etc/passwd and /etc/group files. Changing a user's directory entry will not cause a renaming of the actual directory. It is the user's responsibility to ensure that the /etc/passwd and /etc/group files remain consistent.

Help displays a short informative text on the screen. "?" is equivalent to help. The message is the same one as obtained by invoking ua with the "-h" option.

! escapes to the shell (see sh(C)). If no arguments are given, a shell is invoked which will continue until it receives an end-of-file. Then ua resumes. If arguments are present, a shell is invoked with the "-c" option and the arguments are passed along. Ua resumes immediately thereafter. If csh(C) is desired rather than sh(C), the command !csh should be used.

Quit immediately terminates ua and returns to the system.

Any command which is not understood by ua causes an appropriate message to be displayed. As a side-effect, the working portion of the screen is cleared.

Ua does not distinguish between RETURN and LINE FEED. They may be used interchangably.

If the screen becomes "dirty" for some reason, you can force ua to clear it and redisplay the current contents by transmitting an ASCII "DC2." This is Control-r on most of the currently popular terminals.

Ua understands the Backspace function (as obtained from /etc/termcap). In addition, any time a word is partially formed, the ESCape key will cause the partial word to be discarded and input restarted.

Ua interprets the CANcel key to mean "terminate the current operation." The CANcel key is Control-x on many of the more popular terminals. The CANcel key is more powerful than ESCape, but not so powerful as "interrupt."

Ua will immediately return to the top-level command interpreter upon receipt of an interrupt signal. Such a signal is usually generated via the DEL, RUBOUT or BREAK key.

Ua creates a special user named "standard" in /etc/passwd if one is not already present. This entry is used as the template for installing new users. Thus, if it is desired to have all new users defaulted to the standard XENIX shell (/bin/sh) for the Shell field, it is only necessary to update the Shell field in the "standard" user.

Before adding a new user with a new group, the new group should be added. Otherwise, ua has no way to properly create the new entry in /etc/passwd since it contains group numbers rather than group names.

During program initialization ua copies /etc/passwd and /etc/group to /etc/opasswd and /etc/ogroup, respectively. Thus, if a mistake or disaster occurs during the use of this program, the user may recover the prior state of either or both files.

Files

/etc/passwd	used for login name to user ID conversions
/etc/group	used for group name to group ID conversions
/etc/opasswd	this file is a copy of /etc/passwd before any modifications are made
/etc/ogroup	this file is a copy of /etc/group before any modifications are made
/etc/newuser	directory containing files which will be installed in a new user's account
/etc/termcap	contains terminal attribute descriptions
/tmp/passwd	temporary file
/tmp/group	temporary file
/etc/ua.lock	lock file

See Also

group(F), passwd(F)

Diagnostics

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

Notes

Ua should allow specification of alternate passwd and group files.

Complete consistency checking between the /etc/passwd and /etc/group files is not done. In particular, it is possible to install a user with an unknown group in the passwd file and it is possible to install a group with an unknown user in the group file. The shells and directories specified in the /etc/passwd file are not checked for existence or accessibility.

Ua does not check for duplicated user IDs or duplicated group IDs.

Impossible user IDs and group IDs are permitted.

Impossible or non-existent names may be specified for a user's Directory and Shell fields.

The commands pwcheck (C), grpcheck (C), pwck(1M), and grpck(1M) should be in-corporated into ua.

Name

uname - Prints the current XENIX name.

Syntax

uname [-snrmduvia]

Description

Uname prints the current system name of XENIX on the standard output file. The options cause selected information returned by uname(S) to be printed:

- s Prints the system name (default)
- n Prints the nodename (the nodename may be a name that the system is known by to a communications network)
- r Prints the operating system release
- m Prints original supplier (Manufacturer) of XENIX system
- d Prints OEM (Distributor) for this system
- u Prints user serial number for this system
- i Prints an integer representing the machine type:

  1 Altos 586 or 986
  2 186
  6 486
  7 586T

- a Prints all the above information

See Also

uname(S)

Name

utmp, wtmp – Formats of utmp and wtmp entries.

Description

The files utmp and wtmp hold user and accounting information for use by commands such as *who*(C), *acctcon1* (see *acctcon*(C)), and *login*(M). They have the following structure, as defined by /usr/include/utmp.h:

```
struct utmp
{
        char    ut_line[8];     /* tty name */
        char    ut_name[8];     /* login name */
        long    ut_time;        /* time on */
};
```

Files

/etc/utmp

/usr/adm/wtmp

/usr/include/utmp.h

See Also

acctcon(C), login(M), who(C), write(C)

This page intentionally left blank.

Name

wall - Write to all users

Syntax

/etc/wall

Description

Wall reads its standard input until an end-of-file. It then sends this message, preceded by 'Broadcast Message ... ', to all logged in users.

The sender should be super-user to override any protections the users may have invoked.

Files

/dev/tty*

See Also

mesg(C), write(C)

Diagnostics

'Cannot send to ...' when the open on a user's tty file fails.

Name

who - Who is on the system

Syntax

who [who-file] [am I]

Description

Who, without an argument, lists the login name, terminal name, and login time for each current XENIX user.

Without an argument, who examines the /etc/utmp file to obtain its information. If a file is given, that file is examined. Typically the given file will be /usr/adm/wtmp, which contains a record of all the logins since it was created. The who lists logins, logouts, and crashes since the creation of the wtmp file. Each login is listed with user name, terminal name (with '/dev/' suppressed), and date and time. When an argument is given, logouts produce a similar line without a user name. Reboots produce a line with 'x' in the place of the device name, and a fossil time indicative of when the system went down.

With two arguments, as in 'who am I' (and also 'who are you'), who tells who you are logged in as.

Files

/etc/utmp

See Also

getuid(S), utmp(F)

Control Character Sequences B

CONTENTS

Control character sequences provide the terminal functions described in the chart below. To form a control sequence, press the Control key while pressing the designated alpha key.

CONTROL CHARACTER	FUNCTION
<Control-D>	Interrupts process
<Control-H>	Backspaces and erases
<Control-S>	Stops display (freezes)
<Control-Q>	Releases (restarts) display
<Control-X>	Kills line input
Other special keys:	
Delete/Rubout	Kills the executing program

Appendix C
File Transfer Program

CONTENTS

INTRODUCTION

File transfer programs transfer ASCII text files or binary data files from XENIX-to-XENIX, MP/M-to-XENIX, and XENIX-to-MP/M on Altos computer systems. You should be familiar with XENIX and MP/M before you use these programs.

The programs only transfer files; they do NOT convert MP/M programs to XENIX compatible programs or XENIX programs to MP/M programs.

NOTE

Use the ftp program between Altos computer systems. For transferring files between other XENIX or UNIX computer systems, use the cu or uucp utility. These utilities are described in the XENIX Development System Manual.

Setup Procedures

Before you transfer files,

1. Connect the physical port on each machine via a null modem cable, which is a standard RS232 cable that swaps lines 2 and 3, 4 and 5, and 6 and 20. Refer to the chart below to determine the appropriate port.

SYSTEM	SENDING/RECEIVING OPERATING SYSTEM	DEFAULT PORT
ALTOS 186	XENIX CP/M, MP/M	3
ALTOS 486	XENIX CP/M, MP/M	5
ALTOS 586	XENIX CP/M, MP/M	6
ALTOS 986	XENIX CP/M, MP/M	6 10
ALTOS 8600	XENIX CP/M, MP/M	6 8

You may need to disconnect the printer cable before installing the null modem cable, or install a selector switch.

XENIX can use any available port, but first you must disable it. To disable the XENIX sending/re-

ceiving port(s), become super user, and enter

```
# disable device <CR>
```

where

device = the special file device that transfers files between machines. For example, if you have connected tty2 to the other machine, the device is /dev/tty2. The sending/receiving port numbers don't have to be the same.

NOTE

If your Altos system has WorkNet, do use port 3 for the file transfer program.

If the cable gets disconnected during transmission, wait for the file transfer procedure to stop (takes up to a minute) before restarting on the same port. Otherwise, the first file transfer procedure interferes with the second.

2. Select the same baud rates for both machines. For MP/M 16-bit machines, enter **MPMSETUP.CMD <CR>** and alter the port configuration to set the correct baud rate.

   Altos systems can run at 9600 baud on send and receive. Use the Business Shell port configuration utility to set the correct baud rate.

3. Make sure file names are compatible between systems (you can copy the file and rename it). Files sent from MP/M or CP/M systems to XENIX systems may contain extra spaces. If you enclose the entire filename in quotes, XENIX recognizes it as the intended file name.

Determining Procedures

Refer to the chart below to determine the appropriate file transfer procedure.

SENDING OPERATING SYSTEM	RECEIVING OPERATING SYSTEM	USE PROCEDURE
CP/M - MP/M	XENIX	A
XENIX	XENIX	B
XENIX	CP/M - MP/M	C

THE FILE TRANSFER UTILITY FOR MP/M-TO-XENIX -- PROCEDURE A

Instructions

The File Transfer Program, **FTP86**, resident on both MP/M master distribution diskettes, transfers files to a XENIX system from any 8- or 16-bit Altos Computer System. FTP86 provides full error checking. Correction is accomplished by re-transmission of data blocks.

Follow the setup procedures on page C-2 and C-3.

It does not matter which side, sending or receiving, is started first, as long as both sides are started within one minue of each other.

Start the sending side by entering one of the following commands:

```
ØC>ftp86 filename <CR>

     or

ØC>ftp86 u: filename <CR>
```

where

filename = the name of the file you are transferring.

u: = the drive letter of the destination disk. If no drive letter is specified, the logged disk is the destination disk.

The screen displays the following:

```
File Transfer Program version 3.Ø
Copyright (C) 1982 by Altos Computer Systems
```

The sending side selects the ftp port, and displays an "s" every few seconds until communication is established with the other side.

Start the receiving side of the transfer by using the command format

```
$ ftp [-f device] [-s speed] [name] <CR>
```

where

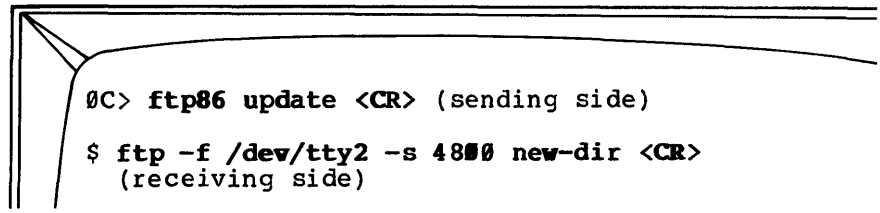
device = the special file device that transfers files between machines. The default device is /dev/tty3 (port 3) on the 186 and /dev/tty6 (port 6) on the 586/986, and /dev/tty5 (port 5) on the 486. If you don't specify the device, omit the -f. Then ftp will use the default device.

speed = transmission speed: 1200, 2400, 4800, or 9600 bits per second. The default is 9600 baud. If you don't specify the speed, omit the -s also. Then ftp will use the default speed.

name = directory, if other than home directory. For example, if you want to transfer the file "update" to your directory "newdir," enter "newdir" as the name.

Do not enter the square brackets ([ ]). They indicate that the enclosed part of the command is optional.

For example, to transfer the file named "update" to the "newdir" directory on the XENIX system, enter



0C> **ftp86 update <CR>** (sending side)

$ **ftp -f /dev/tty2 -s 4800 new-dir <CR>**
(receiving side)

If you do not start procedures within a minute of each other, XENIX will time out and the # prompt reappears. To return to the MP/M prompt, type **<CONTROL-C>**. Then restart the procedures.

The receiving FTP periodically displays a "w" while waiting for the sender to become active.

THE FILE TRANSFER UTILITY FOR XENIX-TO-XENIX -- PROCEDURE B

The XENIX file transfer program, ftp, can transfer files between two Altos Computer Systems running the XENIX operating system.

Instructions

Follow the setup procedures on page C-2 and C-3.

It does not matter which side, sending or receiving, is started first, as long as both sides are started within one minute of each other.

Start the ftp utility by using the following command format on the sending computer:

```
ftp [-f device]  [-s speed]  name
```

where

device = the special file device that transfers files between machines. The default

device is /dev/ftp, which uses port 3 on the 186, port 6 on the 586/986, and port 5 on the 486. The sending/receiving port numbers don't have to be the same. If you don't specify the device, omit the -f also. Then ftp will use the default device.

speed = transmission speed: 1200, 2400, 4800, or 9600 (the default) bits per second. If you don't specify the speed, omit the -s also. Then ftp will use the default speed.

name = the name of the file you are sending.

Do not enter the square brackets ([ ]). They indicate that the enclosed part of the command is optional.

The sending side displays an "s" every few seconds until communication is established with the other side.

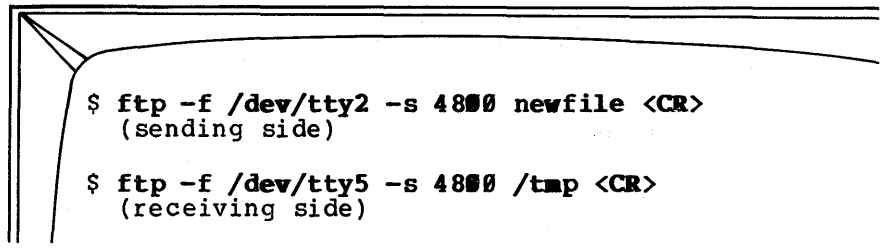
Enter the ftp utility on the receiving computer using the format

```
ftp [-f device] [-s speed] [name]
```

The device can differ from the sending device; however, the speed of the two systems must be the same. Enter the name only if you want to specify a directory for the transferred file other than your home directory.

The receiving side displays a "w" every few seconds. During the file transfer, the ftp utility outputs an "*" after each successful transfer of 128-byte block increments. A "?" is displayed each time a block is retransmitted to overcome a transmission error. If you receive many "?"s, decrease the baud rate.

For example, to transfer the file named "newfile" on the sending XENIX system to the directory "/tmp" on the receiving XENIX system, enter



THE FILE TRANSFER UTILITY FOR XENIX-TO-MP/M -- PROCEDURE C

The XENIX file transfer program, ftp, can transfer files from a XENIX system to an MP/M system.

The XENIX ftp runs on the XENIX system, and the FTP86 runs on the MP/M system during file transfer between XENIX and MP/M.

Instructions

Follow the setup procedures on page C-2 and C-3.

It does not matter which side, sending or receiving, is started first, as long as both sides are started within one minute of each other.

Start the ftp utility by using the following command format on the sending system:

```
ftp [-f device]  [-s speed]  [name]
```

where

device = the special file device that transfers files between machines. If you don't specify the device, omit the -f also. Then ftp will use the default device.

speed = transmission speed: 1200, 2400, 4800, or 9600 bits per second. The default is 9600 baud. If you don't specify the speed, omit the -s also. Then ftp will use the default speed.

name = the name of the file you are sending.

Do not enter the square brackets ([ ]). They indicate that the enclosed part of the command is optional.

For example, to transfer a file named SAMPLE.TXT to the MP/M system, enter

```
$ ftp -f /dev/tty2 -s 4800 SAMPLE.TXT <CR>
```

The sending side displays an "s" every few seconds until communication is established with the other side.

Start the receiving side by entering one of the following commands:

```
0C>FTP86

     or

0C>FTP86 u:
```

where

u: = the drive letter of the destination disk. If no drive letter is specified, the logged disk is the destination disk.

The screen displays the following:

```
File Transfer Program version 3.Ø
Copyright (C) 1982 by Altos Computer Systems
```

The receiving side selects the ftp port, and periodically displays a "w" while waiting for the sender to become active. If the XENIX system times out, the receiving side normally does not exit by itself; type **<Control-C>** to get back to the MP/M prompt.

During the file transfer, the ftp utility outputs an "*" after each successful transfer of 128-byte block increments. A "?" is displayed each time a block is retransmitted to overcome a transmission error. If you receive many "?"s, decrease the baud rate.

Appendix D
Upgrading your Xenix Operating System

CONTENTS

This appendix describes how to upgrade your XENIX operating system. If you are installing XENIX for the first time, use the procedures described in Chapter 1. This appendix also describes how to install a second hard disk.

INTRODUCTION

Before you begin the upgrade procedure, make a copy of each upgrade diskette, and label each diskette by hand.

When you upgrade the system, XENIX

- Preserves system files that you have probably changed (e.g., /etc/passwd, which changes when you run the User Administration program)
- Preserves user files
- Replaces other system files with new files of the same name.

You must be the super user to use this procedure, and other users must be logged off.

UPGRADE PROCEDURE

To upgrade your system, assemble the diskettes you are using for the upgrade, and proceed as follows.

1. Log in as admin and enter admin's password.
2. Shut down the system. To do this from the Business Shell, type **k** to access the System Administration menu. Then type **s** to shut down the system.
3. The shut down program displays

```
Minutes until shutdown? (0 - 15)
```

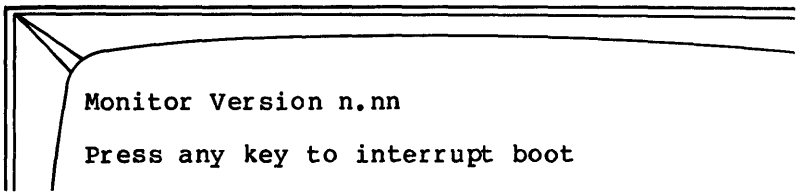
Enter the number of minutes; if no one is on the system, enter

0 <CR>

XENIX broadcasts a message that the system is shutting down, and then displays

```
XENIX will now terminate.
** Normal System Shutdown **
```

4. Press the reset button. You will see a monitor sign-on message; however, your screen may be different from the screen below. Prepare to press any key when you are prompted.



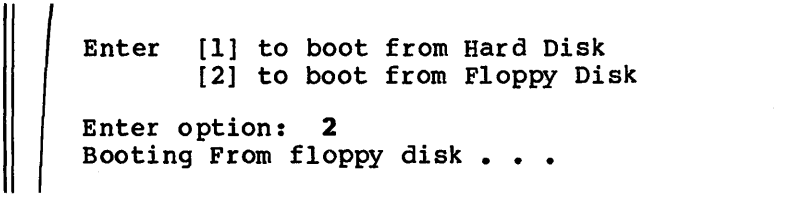
```
Monitor Version n.nn

Press any key to interrupt boot
```

If you press a key in time, you will see a menu (Step 5 shows the first two items of the menu). If not, press the RESET button, and press any key when prompted.

NOTE

Make sure your copy of the XENIX Root File System diskette does not have a write-protect tab on it, so the system can place information on it.

5. When you see the menu below, or one similar to it, insert your copy of the diskette labelled "XENIX Root File System" into the disk drive. Enter 2 to boot from the floppy diskette.

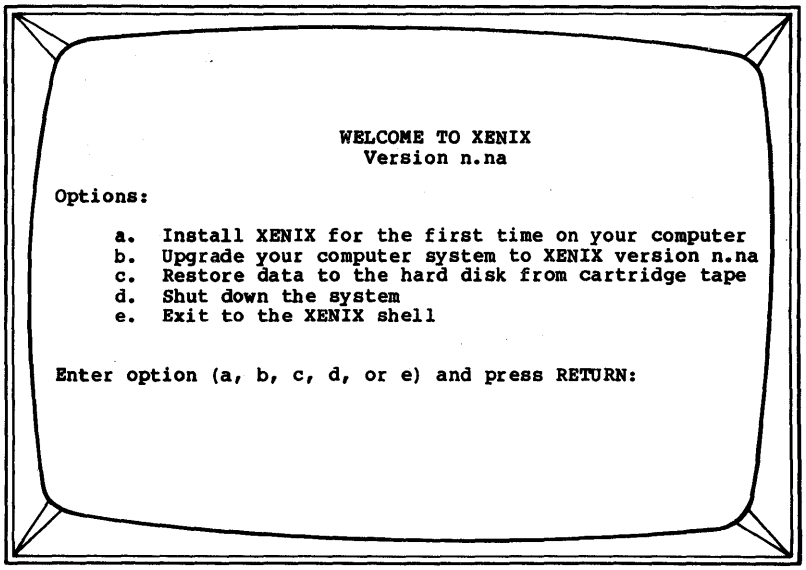


```
Enter  [1] to boot from Hard Disk
       [2] to boot from Floppy Disk

Enter option:  2
Booting From floppy disk . . .
```

After a delay of about 45 seconds, the following message appears.

```
XENIX vn.na

mem = nnnk
```

6. The screen then displays the Welcome to XENIX menu.



Enter **b <CR>**

NOTE

The 486 does not have the restore data from tape option, and the screen adjusts accordingly.

7. XENIX displays messages as it upgrades the system. First it checks the file system. Then the upgrade procedure begins. Next, it saves the local system files. After this, the screen displays

   Remove the "XENIX Root File System" diskette and store it in a safe place.

   Please insert the diskette labelled "XENIX Utilities" and press RETURN.

8. Remove the XENIX Root System diskette and store it in a safe place.

9. Insert your copy of the diskette labelled "XENIX Utilities," and press the Return key.

   The system copies the utilities from the floppy diskette to the hard disk. You will see messages of the form:

   x filename, nnnnn bytes, nn tape blocks

You will also see messages saying that a file has been linked to another file. These messages are for information only.

When this process is finished, the screen displays

> Remove the "XENIX Utilities" diskette and store it in a safe place.

10. Remove the XENIX Utilities diskette and store it. Next, XENIX configures the other system files and restores the local file system.

After the upgrade procedure is complete, the system asks you if you want to do some tasks:

> Change the descriptions of the terminal(s) and/or the printer (configure the ports)
>
> Add or change user accounts.

Lastly, XENIX displays the Options menu.

For instructions on these tasks, please turn to Chapter 4, User Administration.

INSTALLING A SECOND HARD DISK

This section tells you how to set up the software for a second hard disk. You must be the super user for this procedure.

To set up the software for a second hard disk, enter

```
# root <CR>
password:      <CR>
# add.hd <CR>
```

The add.hd command initializes the second hard disk, creates the bad sector table, creates the file system, runs the fsck program, makes the /usr2 directory, and mounts the hard disk. Add.hd adds a line to the /etc/rc file, so that every time the system is set up for multiple users (displays "login:" on other terminals), the second hard disk is mounted.

Upgrading Your XENIX Operating System

D

CONTENTS

This appendix describes how to upgrade your XENIX operating system. If you are installing XENIX for the first time, use the procedures described in Chapter 1.

INTRODUCTION

When you upgrade the system, XENIX

- Preserves system files that you have probably changed (e.g., /etc/passwd, which changes when you run the User Administration program)
- Preserves user files
- Replaces other system files with new files of the same name.

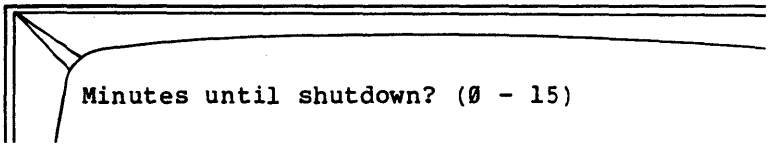
Before you begin, make a copy of each upgrade diskette, and label each diskette by hand.

You must be the super user to use this procedure; other users must be logged off.

UPGRADE PROCEDURE

To upgrade your system, assemble the diskettes you are using for the upgrade, and proceed as follows.

1. Log in as admin and enter admin's password.
2. Shut down the system. To do this from the Business Shell, enter k to access the System Administration menu. Then type s to shut down the system.
3. The shut down program displays



Enter the number of minutes; if no one is on the system, enter

XENIX broadcasts a message that the system is shutting down, and then displays

```
XENIX will now terminate.
** Normal System Shutdown **
```

4. Press the reset button. You will see the following monitor sign-on message. Prepare to press any key when you are prompted.

```
Monitor Version n.nn

Press any key to interrupt boot
```

If you press a key in time, you will see the display in Step 5. If not, press the RESET button, and press any key when prompted.

NOTE

Remove the write-protect tab from the diskette labelled "XENIX Root File System," so the system can place information on it.

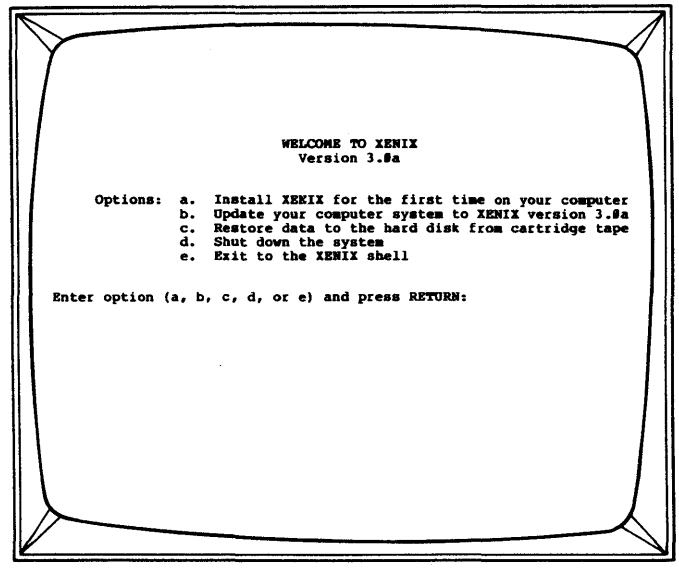
5. When you see the following display, insert your copy of the diskette labelled "XENIX Root File System" into the disk drive. Enter **2** to boot from the floppy diskette.

```
Enter  [1] to boot from Hard Disk
       [2] to boot from Floppy Disk
       [3] to enter Monitor

Enter option:  2
Booting From floppy disk . . .
```

After a delay of about 45 seconds, the following message appears.

```
XENIX vn.na

mem = nnnk
```

6. The screen then displays the Welcome to XENIX menu.



Enter **b <CR>**

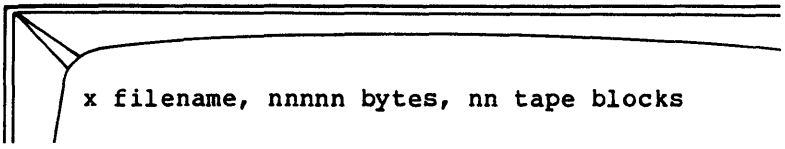
7. XENIX displays messages as it upgrades the system. First it checks the file system. Then the upgrade procedure begins. Next, it saves the local system files. After this, the screen displays

```
Remove the "XENIX Root File System" diskette
and store it in a safe place.

Please insert the diskette labelled "XENIX
Utilities" and press RETURN.
```

8. Remove the XENIX Root System diskette and store it in a safe place.

9. Insert your copy of the diskette labelled "XENIX Utilities," and press the Return key.

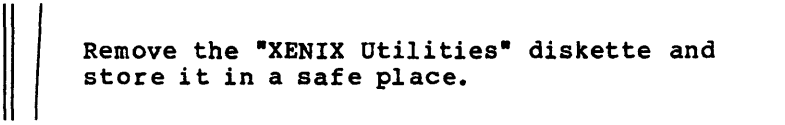
   The system copies the utilities from the floppy diskette to the hard disk. You will see messages of the form:



```
x filename, nnnnn bytes, nn tape blocks
```

   You will also see messages saying that a file has been linked to another file. These messages are for information only.

   When this process is finished, the screen displays



```
Remove the "XENIX Utilities" diskette and
store it in a safe place.
```

10. Remove the XENIX Utilities diskette and store it. Next, XENIX configures the other system files and restores the local file system.

After the upgrade procedure is complete, the system asks you if you want to do some tasks:

   Change the descriptions of the terminal(s) and/or the printer (configure the ports)

   Add or change user accounts.

Lastly, XENIX displays the Options menu.

For instructions on these tasks, please turn to Chapter 4, User Administration.

Using Modems E

CONTENTS

USING MODEMS

The Altos 586/986 XENIX system supports remote communication over telephone lines. You can attach most commercially available asynchronous modems to an Altos system using a standard computer-to-modem cable. Modems that have been used successfully with Altos systems are some models of Racal-Vadic, Cermetek, and Hayes.

When using modems on ports 1 through 6 of the Altos 586/986, make sure that your serial concentrator board is jumpered correctly. Ports 7 through 10 on a serial expander board are already configured correctly for modems. The following table lists jumper positions for ports 1 through 6:

Port	Location	Non-modem jumpers	Modem jumpers
1	E-28	4-6, 1-3	3-4, 5-6
2	E-26	4-6, 1-3	3-4, 5-6
3	E-24	2-4, 3-5	1-3, 4-6
4	E-20	4-6, 1-3	3-4, 5-6
5	E-18	4-6, 1-3	3-4, 5-6
6	E-17	4-6, 1-3	3-4, 5-6

With the jumpers in these positions, the Altos 586/986 will support terminals, printers, or modems on these ports.

Attaching terminals and printers to the Altos 586/986 is a simple operation if you remember that Pin 20 (DTR) must be logic TRUE before any I/O can occur. When attaching a modem to the system, remember that Pin 4 (RTS) must be logic TRUE for "login:" to appear.

To set up a terminal port (ttyn) for modem use, enter

```
$ disable ttyn <CR>
$ /etc/modem ttyn <CR>
$ enable ttyn <CR>
```

where

ttyn = the tty device (n) that has the modem attached.

Note that the disable command isn't necessary if the port is already disabled.

When a user hangs up, the modem command causes him to be logged out and his foreground processes to be terminated.

Execute this command once for every port with a modem attached to your Altos 586/986. Modify the /etc/rc file to include the modem command, so it

will be executed every time you boot the system.

For example, to tell the system that serial port 5 (which is already disabled) is a modem port, enter

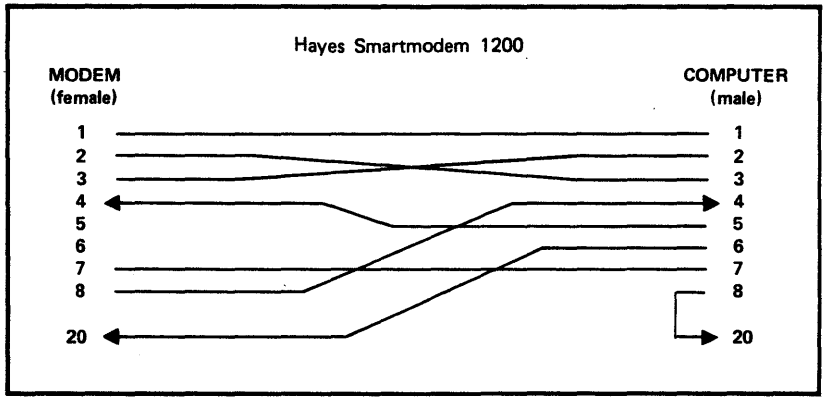
```
$ /etc/modem tty5 <CR>
$ enable tty5 <CR>
```

To unset a modem port and enable it for login, enter

```
$ disable ttyn <CR>
$ /etc/unmodem ttyn <CR>
$ enable ttyn <CR>
```

Note that the disable command isn't necessary if the port is already disabled.

Cable pinouts for the modem interface cable are as follows:



Hayes switch settings are as follows:

	1	2	3	4	5	6	7	8
Dial in or Dial out	up	up	down	down	up	up	up	down

Appendix C of the XENIX Development System Programmer's Guide tells you how to use cu and uucp with modems.

Using Modems E

CONTENTS

USING MODEMS

The Altos 586/986 XENIX system supports remote communication over telephone lines. You can attach most commercially available asynchronous modems to an Altos system using a standard computer-to-modem cable. Modems that have been used successfully with Altos systems are some models of Racal-Vadic, Cermetek, and Hayes.

When using modems on ports 1 through 6 of the Altos 586/986, make sure that your serial concentrator board is jumpered correctly. Ports 7 through 10 on a serial expander board are already configured correctly for modems. The following table lists jumper positions for ports 1 through 6:

Port	Location	Non-modem jumpers	Modem jumpers
1	E-28	4-6, 1-3	3-4, 5-6
2	E-26	4-6, 1-3	3-4, 5-6
3	E-24	2-4, 3-5	1-3, 4-6
4	E-20	4-6, 1-3	3-4, 5-6
5	E-18	4-6, 1-3	3-4, 5-6
6	E-17	4-6, 1-3	3-4, 5-6

With the jumpers in these positions, the Altos 586/986 will support terminals, printers, or modems on these ports.

Attaching terminals and printers to the Altos 586/986 is a simple operation if you remember that Pin 20 (DTR) must be logic TRUE before any I/O can occur. When attaching a modem to the system, remember that Pin 4 (RTS) must be logic TRUE for "login:" to appear.

To set up a terminal port (ttyn) for modem use, enter

```
$ disable ttyn <CR>
$ /etc/modem ttyn <CR>
$ enable ttyn <CR>
```

where

ttyn = the tty device (n) that has the modem attached.

Note that the disable command isn't necessary if the port is already disabled.

When a user hangs up, the modem command causes him to be logged out and his foreground processes to be terminated.

Execute this command once for every port with a modem attached to your Altos 586/986. Modify the /etc/rc file to include the modem command, so it

will be executed every time you boot the system.

For example, to tell the system that serial port 5 (which is already disabled) is a modem port, enter

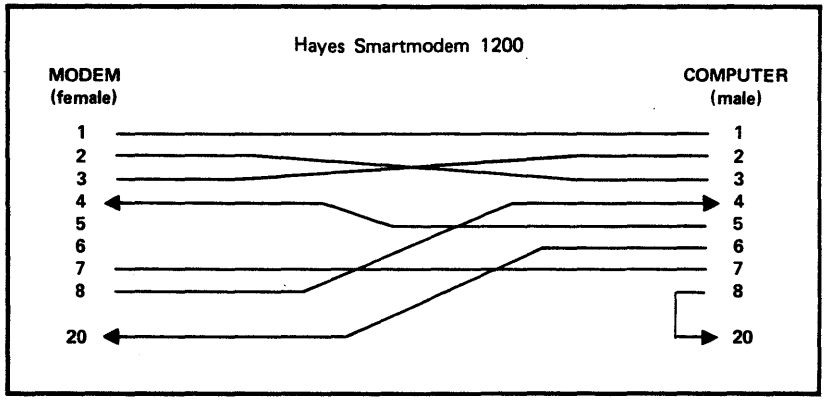
```
$ /etc/modem tty5 <CR>
$ enable tty5 <CR>
```

To unset a modem port and enable it for login, enter

```
$ disable ttyn <CR>
$ /etc/unmodem ttyn <CR>
$ enable ttyn <CR>
```

Note that the disable command isn't necessary if the port is already disabled.

Cable pinouts for the modem interface cable are as follows:



Hayes switch settings are as follows:

	1	2	3	4	5	6	7	8
Dial in or Dial out	up	up	down	down	up	up	up	down

Appendix C of the XENIX Development System Programmer's Guide tells you how to use cu and uucp with modems.

Index

ALTOS 586/986 COMPUTER SYSTEMS
INTRODUCTION TO XENIX- APPENDICES

READER COMMENT FORM

Altos Computer Systems
2641 Orchard Parkway
San Jose, CA 95134

This document has been prepared for use with your Altos Computer System. Should you find any errors or problems in the manual, or have any suggestions for improvement, please return this form to the ALTOS PUBLICATIONS DEPARTMENT. Do include page numbers or section numbers, where applicable.

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

System Model Number_____

Serial Number_____

Document Title_____

Revision Number_____Date_____

Name_____

Company Name_____

Address_____

_____

_____

Printed in U.S.A.
P/N 690-15827-002

2641 Orchard Park Way, San Jose, California 95134
(408) 946-6700, Telex 470642 ALTO UI

February 1985