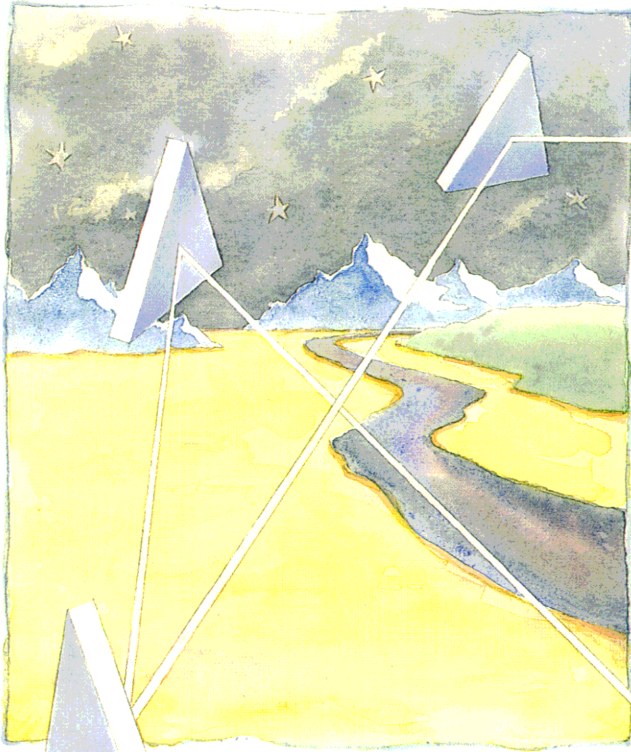




The BSD Command Reference

apollo

BSD Command Reference

Order No. 005800-A01

© Copyright Hewlett-Packard Company 1989. All rights reserved. Reproduction, adaptation, or translation without prior written permission is prohibited, except as allowed under the copyright laws. Printed in USA.

First Printing: July 1988

Last Printing: November 1989

Copyright 1979, 1980, 1983, 1986 Regents of the University of California and 1979, AT&T Bell Laboratories Incorporated.

UNIX is a registered trademark of AT&T in the USA and other countries.

WHILE THE INFORMATION IN THIS PUBLICATION IS BELIEVED TO BE ACCURATE, HEWLETT-PACKARD MAKES NO WARRANTY OF ANY KIND WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Hewlett-Packard shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance or use of this material. Information in this publication is subject to change without notice.

RESTRICTED RIGHTS LEGEND. Use, duplication or disclosure by the Government is subject to restrictions as set forth in subdivision (b) (3) (ii) of the Rights in Technical Data and Computer Software clause at DFARS 52.227-7013. Hewlett-Packard Company, 3000 Hanover Street, Palo Alto, CA 94304.

10 9 8 7 6 5 4 3 2

Preface

The *BSD Command Reference* describes programming utilities available in Domain/OS BSD. This manual is intended for programmers who are familiar with 4.3BSD UNIX software and Domain/OS. It provides neither a general overview of Domain/OS BSD nor details of the implementation of the system. We assume that you are already familiar with the material in *Using Your BSD Environment*.

We have divided the *BSD Command Reference* into three sections:

Commands Section 1 provides reference material on user commands.

Games Section 6 provides reference material on games.

Miscellaneous Section 7 provides miscellaneous useful information.

Each section consists of independent entries of a page or so each. The name of the entry is in the upper corners of its pages, together with the section number, and sometimes a letter characteristic of a class. For example, the **ftp** command is 1C. Each section begins with **intro(N)**, followed by **domain(N)**, where *N* is the number of the section. Entries thereafter appear in alphabetical order.

Some entries may describe several features. In such cases, the entry may appear only once, alphabetized under its “primary” name, the name that appears at the upper corners of each manual page.

Entries with Domain/OS BSD (as contrasted with a simple BSD) centered at the top of the page describe features unique to Domain/OS BSD. Each section contains an entry with the name **domain** that provides an overview of the unique features in that section.

We use the convention **name(N)** to cite entry **name** in section *N* of this and other manuals. References to sections other than 1, 6, or 7 mean that **name** is contained in another manual. The *BSD Programmer's Reference* contains Sections 2, 3, 4, and 5. *Managing BSD System Software* includes Sections 1M and 7.

All entries are based on a common format, not all of whose parts always appear:

NAME	Gives the name of the feature and briefly states its purpose.
SYNOPSIS	Summarizes the use of the feature being described. In the case of system calls and subroutines, this summary usually specifies header files (by way of the appropriate #include <i><file.h></i> preprocessor statement) containing definitions needed by the call or subroutine. This summary also usually contains a set of declarations as they might appear in a C-language function header defining the call or subroutine.
DESCRIPTION	Describes the feature.
EXAMPLE(S)	Gives example(s) of usage, where appropriate.
FILES	Gives the filenames that are built into the feature.
SEE ALSO	Gives pointers to related information.
DIAGNOSTICS	Discusses the diagnostic indications that may be produced. Messages that are intended to be self-explanatory are not listed.
NOTES	Gives generally “helpful hints” about the use of the feature.
WARNINGS	Points out potential pitfalls.
BUGS	Gives known bugs and sometimes deficiencies.
CAVEATS	Gives details of the implementation that might affect usage.

A “Table of Contents” and a “Permuted Index” derived from the Table of Contents precede Section 1. The Permuted Index is a list of keywords, given in the second of three columns, together with the context in which each keyword is found. Keywords are either topical keywords or the names of manual entries. Entries are identified with their section numbers shown in parentheses. This is important because there is considerable duplication of names among the sections, arising principally from components that exist only to exercise a particular system call. The right column lists the name of the manual page on which each keyword may be found. The left column contains useful information about the keyword.

Online Access

We deliver a machine-readable version of this manual (and Section 8) in the files */bsd4.3/usr/man/cat[1678]/name.[1678]class*, where *name* is that of the feature documented, [167] is either 1, 6, or 7, depending upon the section, and *class* (C for communication, G for graphics, etc.) may or may not appear.

If you have installed these files on your workstation, or you have links from your workstation to one where these files are installed, you may access them by way of the **man(1)** command. To read about **man**, type

% man 1 man

or refer to **man(1)** in this book.

Related Manuals

The file `/install/doc/apollo/os.v.latest software release number__manuals` lists current titles and revisions for all available manuals.

For example, at Software Release 10.2 (SR10.2) you may refer to the file `/install/doc/apollo/os.v.10.2__manuals` to check that you are using the correct version of manuals. You may also want to use this file to check that you have ordered all of the manuals that you need.

The same information is available online. In the UNIX environment, type **man manuals**. In the Aegis environment, **help manuals**.

Refer to the *Domain Documentation Quick Reference* (002685) and the *Domain Documentation Master Index* (011242) for a complete list of related documents.

For introductory information about the Domain/OS system and details about using the BSD environment, refer to the following documents:

- *Getting Started with Domain/OS* (2348)
- *Using Your BSD Environment* (11020)
- *BSD User's Manual*, Volumes I and II (017271 and 017623)
- *Domain Display Manager Command Reference* (11418)

For more information on programming in the Domain/OS BSD environment, refer to the following documents:

- *Domain/OS Call Reference*, Volumes 1 and 2 (7196 and 12888)
- *Domain/OS Programming Environment Reference* (11010)
- *Domain Binder and Librarian Reference* (4977)
- *Domain C Language Reference* (2093)
- *BSD UNIX Programmer's Manual* (017272)

- *BSD Programmer's Reference* (5801)
- *Managing BSD System Software* (10853)

Problems, Questions, and Suggestions

We appreciate comments from the people who use our system. To make it easy for you to communicate with us, we provide the Apollo Product Reporting (APR) system for comments related to hardware, software, and documentation. By using this formal channel you make it easy for us to respond to your comments.

You can get more information about how to submit an APR by consulting the appropriate Command Reference manual for your environment (Aegis, BSD, or SysV). Refer to the **mkapr** shell command description. You can view the same description online by typing:

\$ man 1 mkapr (in the SysV environment)

% man 1 mkapr (in the BSD environment)

\$ help mkapr (in the Aegis environment)

Alternatively, you may use the Reader's Response form at the back of this manual to submit comments about the manual.

Documentation Conventions

This manual uses the following symbolic conventions:

literal values	Bold words or characters in formats and command descriptions represent commands or keywords that you must use literally. Bold words in text indicate the first use of a new term. Filenames and pathnames are also in bold.
<i>user-supplied values</i>	Placeholders for symbols that you must supply are printed in italics. For example, the names chosen for call arguments appear in italics.
sample user input	In samples, information that the user enters appears in bold.
examples	Examples of program code and program output appear in this typeface.

[]	Square brackets enclose optional items in formats and command descriptions.
{ }	Braces enclose a list from which you must choose an item in formats and command descriptions.
	A vertical bar separates items in a list of choices.
< >	Angle brackets enclose the name of a key on the keyboard.
CTRL/ ^	The notation CTRL/ or ^ followed by the name of a key indicates a control character sequence. Hold down <CTRL> while you press the key.
...	Ellipses mean that the previous argument-prototype may be repeated.
:	Vertical ellipsis points mean that irrelevant parts of a figure or example have been omitted.
-	An argument beginning with a dash (“-”) usually means that it is an option-specifying argument used by the command itself, even if it appears in a position where a file name could appear. Therefore, it is unwise to have files whose names begin with “-”.
————— ☐ —————	This symbol indicates the end of a section.

Contents

Reference Pages

1: Commands and Application Programs

addbib(1).....	create or extend bibliographic database
admin(1).....	create and administer SCCS files
apply(1).....	apply a command to a set of arguments
apropos(1).....	locate commands by keyword lookup
ar(1).....	archive and library maintainer
at(1).....	execute commands later
atq(1).....	print the queue of jobs waiting to be run
atrm(1).....	remove jobs spooled by
awk(1).....	pattern scanning and processing language
basename(1).....	strip filename affixes
bc(1).....	arbitrary-precision arithmetic language
bdf2snf(1).....	BDF to SNF font compiler for X11
bdiff(1).....	big diff
biff(1).....	be notified if mail arrives and who it is from
binmail(1).....	send or receive mail among users
bldt(1).....	display time operating system was built
cal(1).....	print calendar
calendar(1).....	reminder service
cat(1).....	catenate and print
cb(1).....	C program beautifier
cc(1).....	C compiler
cd(1).....	change working directory
cdc(1).....	change the delta commentary of an SCCS delta
chac1(1).....	change access control list
checknr(1).....	check nroff/troff files
chgrp(1).....	change group
chmod(1).....	change mode
clear(1).....	clear terminal screen
cmp(1).....	compare two files
coffdump(1).....	dump selected parts of an object file
col(1).....	filter reverse line feeds
colcrt(1).....	filter nroff output for CRT previewing
colrm(1).....	remove columns from a file

comb(1).....	combine SCCS deltas
comm(1).....	select or reject lines common to two sorted files
compact(1).....	compress and uncompress files, and then cat them
compress(1).....	compress and expand data
cp(1).....	copy files
cpacl(1).....	copy access control list
cpscr(1).....	copy the current display to a file
crddf(1).....	create, display, or modify a device descriptor file
crp(1).....	create a process on a remote node
crpad(1).....	create a transcript pad and window
crty(1).....	create a new type
crtyobj(1).....	create a type object module for binding
csh(1).....	a shell (command interpreter) with C-like syntax
ctags(1).....	create a tags file
cvt_font(1).....	convert fonts from pre-SR10 to SR10 format
cvtname(1).....	convert pathnames between upper and lowercase and preserve colons
cvtrgy(1).....	convert registry between SR9.x and SR10 formats
date(1).....	print and set the date
dbacl(1).....	Domain/Dialogue-based access control list editor
dbx(1).....	source code debugger
dc(1).....	desk calculator
dd(1).....	convert and copy a file
dde(1).....	Domain Distributed Debugging Environment
delta(1).....	make a delta (change) to an SCCS file
deroff(1).....	remove nroff , troff , tbl , and eqn constructs
df(1).....	disk free
diction(1).....	print wordy sentences; thesaurus for diction
diff(1).....	differential file and directory comparator
diff3(1).....	three-way differential file comparison
dlty(1).....	delete a type
dm(1).....	dm/commands - Display Manager Commands
dmwin(1).....	create a DM transcript/input pad pair and execute a program in it
domain(1).....	Domain/OS-specific commands and extensions
dspst(1).....	display process status graphically
dstdump(1).....	dump a COFF file's debugging information to
du(1).....	summarize disk usage
echo(1).....	echo arguments
ed(1).....	text editor
edfont(1).....	edit a character font
edmtdesc(1).....	edit magtape descriptor file
emt(1).....	emulate a dumb terminal
eqn(1).....	format mathematical text
error(1).....	analyze and disperse compiler error messages
esa(1).....	display address of external symbol
ex(1).....	text editor
expand(1).....	expand tabs to spaces, and vice versa
expr(1).....	evaluate arguments as an expression
f77(1).....	FORTAN 77 compiler

false(1) provide truth values
file(1) determine file type
find(1) find files
finger(1) user information lookup program
fmt(1) simple text formatter
fold(1) fold long lines for finite-width output device
fpr(1) print FORTRAN file
from(1) who is my mail from?
fsplit(1) split a multiroutine FORTRAN file into individual files
fst(1) print fault status information
ftp(1c) ARPANET file transfer program
get(1) get a version of an SCCS file
gprof(1) display call graph profile data
graph(1g) draw a graph
grep(1) search a file for a pattern
groups(1) show group memberships
head(1) give the first few lines of a file
help(1) ask for SCCS help
hostid(1) set or print identifier of current host system
hostname(1) set or print name of current host system
hpc(1) program counter histogram
indent(1) indent and format C program source
inlib(1) install a user-supplied library
install(1) install binaries
intm(1) install a type manager
intro(1) introduction to commands
inty(1) install a new type
iso(1) convert files to ISO format
join(1) relational database operator
kbm(1) set/display keyboard characteristics
kill(1) terminate or signal a process
ksh(1) the Korn shell command programming language
las(1) list objects mapped into the address space
last(1) indicate last logins of users and teletypes
lastcomm(1) show last commands executed in reverse order
lbr2ar(1) convert lbr libraries to SR10 archive libraries
lcm(1) load a color map
ld(1) link editor for common object files
leave(1) remind you when you have to leave
lex(1) generator of lexical analysis programs
lint(1) a C program verifier
llib(1) list installed libraries
llkob(1) list locked objects
ln(1) make links
logger(1) make entries in the system log
login(1) sign on
look(1) find lines in a sorted list
lookbib(1) build inverted index for a

lorder(1) find ordering relation for an object library
 lpq(1) spool queue examination program
 lpr(1) print files offline
 lprm(1) remove jobs from the line-printer spooling queue
 lptest(1) generate lineprinter ripple pattern
 ls(1) list contents of directory
 lsacl(1) list access control list
 lty(1) list installed types
 m4(1) macro processor
 mail(1) send and receive mail
 make(1) update and regenerate programs or groups of programs
 man(1) display reference manual information
 mesg(1) permit or deny messages
 mk(1) maintain (make) related files
 mkapr(1) make an Apollo product report
 mkdir(1) make a directory
 mkstr(1) create an error message file by massaging C source
 more(1) file perusal filter for CRT viewing
 msgs(1) system messages and junk mail program
 mt(1) magnetic tape manipulating program
 mv(1) move or rename files
 nawk(1) pattern scanning and processing language
 netstat(1) show network status
 newaliases(1) rebuild the database for the mail aliases file
 nice(1) run a command at a changed priority (**sh** only)
 nm(1) print name list
 nroff(1) text formatting
 obj2coff(1) convert OBJ format modules to COFF format modules
 od(1) octal, decimal, hex, ASCII dump
 pagesize(1) print system page size
 pax(1) portable archive exchange
 plot(1g) graphics filters
 pr(1) print file
 prf(1) queue a file for printing by Domain/OS Aegis print spooler
 print(1) **pr** to the line printer
 printenv(1) print out the environment
 prism(1) get processor type truth value for prism
 prof(1) display profile data
 prs(1) print an SCCS file
 ps(1) process status
 ptx(1) permuted index
 pwd(1) working directory name
 ranlib(1) convert archives to random libraries
 ratfor(1) rational FORTRAN dialect
 rbak(1) restore or index a magnetic media backup file
 rcp(1c) remote file copy
 rdist(1) remote file distribution program
 refer(1) find and insert literature references in documents

rev(1) reverse lines of a file
 rlogin(1c) remote login
 rm(1) remove (unlink) files or directories
 rmail(1) handle remote mail received via **uucp**
 rmdel(1) remove a delta from an SCCS file
 rmdir(1) remove (unlink) directories or files
 roffb(1) run off bibliographic database
 rsh(1c) remote shell
 ruptime(1c) show host status of local machines
 rwho(1c) who's logged in on local machines
 rwmt(1) read/write foreign magtapes
 sact(1) print current SCCS file editing activity
 sccs(1) front end for the SCCS subsystem
 sccsdiff(1) compare two versions of an SCCS file
 scrattr(1) screen attributes
 script(1) make typescript of terminal session
 scrto(1) set/show screen timeout
 sed(1) stream editor
 sh(1) the Bourne shell command language
 size(1) size of an object file
 sleep(1) suspend execution for an interval
 soelim(1) eliminate .so's from nroff input
 sort(1) sort or merge files
 sortbib(1) sort bibliographic database
 spell(1) find spelling errors
 spline(1g) interpolate smooth curve
 split(1) split a file into pieces
 start_csh(1) start a log-in C shell
 start_sh(1) start a log-in Bourne shell
 stcode(1) translate status code value to text message
 stk_font_gen(1) create a custom stroke font
 strings(1) find the printable strings in an object file
 strip(1) remove symbols and relocation bits
 struct(1) structure FORTRAN programs
 stty(1) set terminal options
 style(1) analyze surface characteristics of a document
 su(1) substitute user ID temporarily
 sum(1) sum and count blocks in a file
 swapul(1) rearrange underlining
 systype(1) display version stamp
 tabs(1) set terminal tabs
 tail(1) deliver the last part of a file
 talk(1) talk to another user
 tar(1) tape archiver
 tb(1) print process traceback
 tbl(1) format tables for nroff or troff
 tcopy(1) copy a mag tape
 tee(1) pipe fitting

telnet(1c).....	user interface to the TELNET protocol
test(1).....	condition command
tftp(1c).....	trivial file transfer protocol
time(1).....	time a command
tip(1c).....	connect to a remote system
touch(1).....	update date last modified of a file
tpm(1).....	set/display touchpad and mouse characteristics
tr(1).....	translate characters
tr_font(1).....	transliterate characters within a font
troff(1).....	text formatting and typesetting
trty(1).....	test a type trait.
true(1).....	provide truth values
ts(1).....	display the module name and time stamp
tset(1).....	terminal dependent initialization
tsort(1).....	topological sort
tty(1).....	get terminal name
tz(1).....	set or display system time zone
ul(1).....	do underlining
unget(1).....	undo a previous get of an SCCS file
unifdef(1).....	remove ifdef 'ed lines
uniq(1).....	report repeated lines in a file
units(1).....	conversion program
uptime(1).....	show how long system has been up
users(1).....	compact list of users who are on a node
uucp(1c).....	UNIX-to-UNIX system copy
uencode(1c).....	encode/decode a binary file for transmission via mail
uustat(1).....	uucp status inquiry and job control
uuto(1).....	public UNIX-to-UNIX system file copy
uux(1).....	UNIX-to-UNIX system command execution
uwm(1).....	uwm - a window manager for X
val(1).....	validate SCCS file
vc(1).....	version control
ver(1).....	change the version of shell commands
vgrind(1).....	grind nice listings of programs
vi(1).....	screen oriented (visual) display editor based on ex
vsiz(1).....	set/display VT100 window settings
vt100.....	VT100 terminal emulator
wait(1).....	await completion of process
wall(1).....	write to all users
wbak(1).....	create a magnetic media backup file
wc(1).....	word count
what(1).....	show what versions of object modules were used to construct a file
whatis(1).....	describe what a command is
whereis(1).....	locate source, binary, and/or manual for program
which(1).....	locate a program file, including aliases and paths (cs h only)
who(1).....	who is on the system
whoami(1).....	print effective current user id
whois(1).....	DARPA Internet username directory service

write(1)..... write to another user
 X(1).....a portable, network transparent window system
 Xapollo(1)..... Apollo Domain/X11 shared X window system server
 xdm(1)..... execute a DM command from the shell
 xownroot(1)..... switch between X and DM ownership of the root
 Xserver(1).....X Window System server
 xset(1).....xset - user preference utility for X
 xstr(1)..... extract strings from C programs to implement shared strings
 xterm(1)..... terminal emulator for X
 yacc(1)..... yet another compiler-compiler
 yes(1)..... be repetitively affirmative

6: Games

arithmetic(6)..... provide drill in number facts
 backgammon(6)..... the game of backgammon
 banner(6)..... print large banner on printer
 battlestar(6)..... a tropical adventure game
 bcd(6)..... convert to antique media
 bgcolor(6)..... make interesting background colors
 bj(6)..... the game of blackjack
 boggle(6)..... play the game of boggle
 btlfortune(6)..... print a random comment
 btlgammon(6)..... the game of backgammon
 btlhangman(6)..... guess the word
 canfield(6)..... the solitaire card game canfield
 craps(6)..... the game of craps
 cribbage(6)..... the card game cribbage
 dmoire(6)..... Domain/Dialogue-based moire generator
 domain(6)..... Domain/OS-specific games
 factor(6)..... factoring program
 fish(6)..... play "Go Fish"
 flake(6)..... induce terminal dandruff
 fortune(6)..... print a random, hopefully interesting, adage
 hangman(6)..... Computer version of the game hangman
 hunt(6)..... a multi-player multi-terminal game
 intro(6)..... introduction to games
 mastermind(6)..... Mastermind guessing game
 maze(6)..... generate a maze
 melt(6)..... "melt" the screen
 mille(6)..... play Mille Bournees
 monop(6)..... Monopoly game
 moo(6)..... guessing game
 number(6)..... convert Arabic numerals to English
 primes(6)..... print prime numbers
 puzzle(6)..... puzzle game
 quiz(6)..... test your knowledge
 rain(6)..... animated raindrops display
 random(6)..... random number generator

revscr(6)	reverse screen
robots(6)	fight off villainous robots
sail(6)	multi-user wooden ships and iron men
scramble(6)	turn your screen into a scramble puzzle
snake(6)	display chase game
strfile(6)	create a random access file for storing strings
teachgammon(6)	teach the game of backgammon
trek(6)	trekkie game
ttt(6)	tic-tac-toe
vine(6)	grow vines
worm(6)	play the growing worm game
worms(6)	animate worms on a display terminal
wump(6)	the game of hunt-the-wumpus

7: Miscellaneous

acl(7)	details about access control lists (ACLs)
ascii(7)	map of ASCII character set
domain(7)	miscellaneous Domain/OS features
environ(7)	user environment variables
eqnchar(7)	special character definitions for eqn
hier(7)	file system hierarchy
intro(7)	miscellaneous useful information pages
limits(7)	Domain/OS limits
mailaddr(7)	mail addressing description
man(7)	macros to typeset manual
me(7)	macros for formatting papers
ms(7)	text formatting macros
term(7)	conventional names for terminals

PERMUTED INDEX

f77: FORTRAN	77 compiler	f77(1)
chac1: change	access control list	chac1(1)
cpac1: copy	access control list	cpac1(1)
dbac1: Domain/Dialogue-based	access control list editor	dbac1(1)
lsac1: list	access control list	lsac1(1)
ac1: details about	access control lists (ACLs)	ac1(7)
/unstr: create a random	access file for storing/	strfile(6)
control lists (ACLs)	ac1: details about access	ac1(7)
about access control lists	(ACLs) ac1: details	ac1(7)
current SCCS file editing	activity sac1: print	sac1(1)
random, hopefully interesting,	adage fortune: print a	fortune(6)
bibliographic database	addbib: create or extend	addbib(1)
esa: display	address of external symbol	esa(1)
list objects mapped into the	address space las:	las(1)
SCCS files	admin: create and administer	admin(1)
admin: create and	administer SCCS files	admin(1)
battlestar: a tropical	adventure game	battlestar(6)
file for printing by Domain/OS	Aegis print spooler/queue a	prf(1)
yes: be repetitively	affirmative	yes(1)
basename: strip filename	affixes	basename(1)
/a program file, including	aliases and paths (csh only)	which(1)
the database for the mail	aliases file /rebuild	newaliases(1)
lex: generator of lexical	analysis programs	lex(1)
error messages error:	analyze and disperse compiler	error(1)
characteristics of a/ style:	analyze surface	style(1)
/locate source, binary,	and/or manual for program	whercis(1)
terminal worms:	animate worms on a display	worms(6)
rain:	animated raindrops display	rain(6)
bcd: convert to	antique media	bed(6)
window system server Xapollo:	Apollo Domain/X11 "shared" X	Xapollo(1)
mkapr: make an	Apollo product report	mkapr(1)
arguments apply:	apply a command to a set of	apply(1)
set of arguments	apply: apply a command to a	apply(1)
keyword lookup	apropos: locate commands by	apropos(1)
maintainer	ar: archive and library	ar(1)
number: convert	Arabic numerals to English	number(6)
language bc:	arbitrary-precision arithmetic	bc(1)
ar:	archive and library maintainer	ar(1)
pax: portable	archive exchange	pax(1)
convert lbr libraries to SR10	archive libraries lbr2ar:	lbr2ar(1)
tar: tape	archiver	tar(1)
ranlib: convert	archives to random libraries	ranlib(1)
apply a command to a set of	arguments apply:	apply(1)
expr: evaluate	arguments as an expression	expr(1)
echo: echo	arguments	echo(1)
bc: arbitrary-precision	arithmetic language	bc(1)
number facts	arithmetic: provide drill in	arithmetic(6)
ftp:	ARPANET file transfer program	ftp(1C)
biff: be notified if mail	arrives and who it is from	biff(1)
expr: evaluate arguments	as an expression	expr(1)
ascii: map of	ASCII character set	ascii(7)
od: octal, decimal, hex,	ASCII dump	od(1)
set	ascii: map of ASCII character	ascii(7)
help:	ask for SCCS help	help(1)

waiting to be run	atq: print the queue of jobs	atq(1)
	atrm: remove jobs spooled by	atrm(1)
scrattr: screen	attributes	scrattr(1)
wait:	await completion of process	wait(1)
processing language	awk: pattern scanning and	awk(1)
backgammon: the game of	backgammon	backgammon(6)
btgammon: the game of	backgammon	btgammon(6)
teachgammon: teach the game of	backgammon	teachgammon(6)
backgammon	backgammon: the game of	backgammon(6)
bgcolor: make interesting	background colors	bgcolor(6)
or index a magnetic media	backup file rbak: restore	rbak(1)
wbak: create a magnetic media	backup file	wbak(1)
banner: print large	banner on printer	banner(6)
printer	banner: print large banner on	banner(6)
(visual) display editor	based on ex /screen oriented	vi(1)
affixes	basename: strip filename	basename(1)
adventure game	battlestar: a tropical	battlestar(6)
arithmetic language	bc: arbitrary-precision	bc(1)
	bcd: convert to antique media	bcd(6)
X11 bdfstosnf -	BDF to SNF font compiler for	bdfstosnf(1)
compiler for X11	bdfstosnf - BDF to SNF font	bdfstosnf(1)
	bdiff: big diff	bdiff(1)
cb: C program	beautifier	cb(1)
background colors	bgcolor: make interesting	bgcolor(6)
addbib: create or extend	bibliographic database	addbib(1)
roffbib: run off	bibliographic database	roffbib(1)
sortbib: sort	bibliographic database	sortbib(1)
arrives and who it is from	biff: be notified if mail	biff(1)
install: install	binaries	install(1)
whereis: locate source,	binary, and/or manual for/	whereis(1)
via mail /encode/decode a	binary file for transmission	uuencode(1C)
a type object module for	binding crtyobj: create	crtyobj(1)
among users	binmail: send or receive mail	binmail(1)
remove symbols and relocation	bits strip:	strip(1)
	bj: the game of blackjack	bj(6)
blackjack		bj(6)
system was built	bldt: display time operating	bldt(1)
sum: sum and count	blocks in a file	sum(1)
boggle: play the game of	boggle	boggle(6)
boggle	boggle: play the game of	boggle(6)
sh: the	Bourne shell command language	sh(1)
start_sh: start a log-in	Bourne shell	start_sh(1)
mille: play Mille	Bournes	mille(6)
comment	bulfortune: print a random	bulfortune(6)
backgammon	btgammon: the game of	btgammon(6)
	btlangman: guess the word	btlangman(6)
indxbib, lookbib:	build inverted index for a	lookbib(1)
time operating system was	built bldt: display	bldt(1)
cc:	C compiler	cc(1)
cb:	C program beautifier	cb(1)
indent: indent and format	C program source	indent(1)
lint: a	C program verifier	lint(1)
xstr: extract strings from	C programs to implement shared/	xstr(1)
start_csh: start a log-in	C shell	start_csh(1)
message file by massaging	C source /create an error	mkstr(1)
	cal: print calendar	cal(1)

dc: desk	calculator	dc(1)
cal: print	calendar	cal(1)
	calendar: reminder service	calendar(1)
gprof: display	call graph profile data	gprof(1)
the solitaire card game	canfield canfield, cfscores:	canfield(6)
solitaire card game canfield	canfield, cfscores: the	canfield(6)
cfscores: the solitaire	card game canfield canfield,	canfield(6)
cribbage: the	card game cribbage	cribbage(6)
	cat: catenate and print	cat(1)
and uncompress files, and then	cat them /ccat: compress	compact(1)
cat:	catenate and print	cat(1)
	cb: C program beautifier	cb(1)
	cc: C compiler	cc(1)
files/ compact, uncompact,	ccat: compress and uncompress	compact(1)
	cd: change working directory	cd(1)
commentary of an SCCS delta	cdc: change the delta	cdc(1)
game canfield canfield,	cfscores: the solitaire card	canfield(6)
list	chac1: change access control	chac1(1)
delta: make a delta	(change) to an SCCS file	delta(1)
/nohup: run a command at a	changed priority (sh only)	nice(1)
eqnchar: special	character definitions for eqn	eqnchar(7)
edfont: edit a	character font	edfont(1)
ascii: map of ASCII	character set	ascii(7)
kbm: set/display keyboard	characteristics	kbm(1)
style: analyze surface	characteristics of a document	style(1)
set/display touchpad and mouse	characteristics tpm:	tpm(1)
	characters	tr(1)
tr_font: transliterate	characters within a font	tr_font(1)
snake, sncore: display	chase game	snake(6)
checknr:	check nroff/troff files	checknr(1)
text eqn, neqn,	checkeq: format mathematical	eqn(1)
files	checknr: check nroff/troff	checknr(1)
	chgrp: change group	chgrp(1)
	chmod: change mode	chmod(1)
	clear: clear terminal screen	clear(1)
clear:	clear terminal screen	clear(1)
(command interpreter) with	C-like syntax csh: a shell	csh(1)
	cmp: compare two files	cmp(1)
dbx: source	code debugger	dbx(1)
stcode: translate status	code value to text message	stcode(1)
information/ dsdump: dump a	COFF file's debugging	dsdump(1)
convert OBJ format modules to	COFF format modules obj2coff:	obj2coff(1)
of an object file	coffdump: dump selected parts	coffdump(1)
	col: filter reverse line feeds	col(1)
for CRT previewing	colcrt: filter nroff output	colcrt(1)
and lowercase and preserve	colons /between upper	cvtname(1)
lcm: load a	color map	lcm(1)
make interesting background	colors bgcolor:	bgcolor(6)
file	colrm: remove columns from a	colrm(1)
colrm: remove	columns from a file	colrm(1)
	comb: combine SCCS deltas	comb(1)
comb:	combine SCCS deltas	comb(1)
common to two sorted files	comm: select or reject lines	comm(1)
(sh only) nice, nohup: run a	command at a changed priority	nice(1)
uux: UNIX-to-UNIX system	command execution	uux(1)
xdmc: execute a DM	command from the shell	xdmc(1)

C-like syntax	csh: a shell (command interpreter) with	csh(1)
whatis: describe what a	command is	whatis(1)
sh: the Bourne shell	command language	sh(1)
ksh: the Korn shell	command programming language	ksh(1)
test: condition	command	test(1)
time: time a	command	time(1)
apply: apply a	command to a set of arguments	apply(1)
domain: Domain/OS-specific	commands and extensions	domain(1)
apropos: locate	commands by keyword lookup	apropos(1)
dm/commands - Display Manager	Commands .if ' . dm(1) v'2'	dm(1)
order lastcomm: show last	commands executed in reverse	lastcomm(1)
intro: introduction to	commands	intro(1)
at: execute	commands later	at(1)
change the version of shell	commands ver:	ver(1)
bitfortune: print a random	comment	bitfortune(6)
cdc: change the delta	commentary of an SCCS delta	cdc(1)
ld: link editor for	common object files	ld(1)
comm: select or reject lines	common to two sorted files	comm(1)
on a node users:	compact list of users who are	users(1)
compress and uncompress/	compact, uncompact, ccac:	compact(1)
file and directory	comparator diff: differential	diff(1)
cmp:	compare two files	cmp(1)
SCCS file scsdiff:	compare two versions of an	scsdiff(1)
three-way differential file	comparison diff3:	diff3(1)
cc: C	compiler	cc(1)
error: analyze and disperse	compiler error messages	error(1)
f77: FORTRAN 77	compiler	f77(1)
bdfiosnf - BDF to SNF font	compiler for X11	bdfiosnf(1)
yacc: yet another	compiler-compiler	yacc(1)
wait: await	completion of process	wait(1)
compress, uncompress, zcat:	compress and expand data	compress(1)
and/ compact, uncompact, ccac:	compress and uncompress files,	compact(1)
compress and expand data	compress, uncompress, zcat:	compress(1)
hangman hangman:	Computer version of the game	hangman(6)
test:	condition command	test(1)
tip, cu:	connect to a remote system	tip(1C)
of object modules were used to	construct a file /versions	what(1)
nroff, troff, tbl, and eqn	constructs deroff: remove	deroff(1)
ls: list	contents of directory	ls(1)
chacl: change access	control list	chacl(1)
cpacl: copy access	control list	cpacl(1)
Domain/Dialogue-based access	control list editor dbacl:	dbacl(1)
lsacl: list access	control list	lsacl(1)
acl: details about access	control lists (ACLs)	acl(7)
uucp status inquiry and job	control uustat:	uustat(1)
vc: version	control	vc(1)
terminals term:	conventional names for	term(7)
units:	conversion program	units(1)
dd:	convert and copy a file	dd(1)
English number:	convert Arabic numerals to	number(6)
libraries ranlib:	convert archives to random	ranlib(1)
iso:	convert files to ISO format	iso(1)
SR10 format cvt_font:	convert fonts from pre-SR10 to	cvt_font(1)
archive libraries lbr2ar:	convert lbr libraries to SR10	lbr2ar(1)
COFF format modules obj2coff:	convert OBJ format modules to	obj2coff(1)
upper and lowercase/ cvtname:	convert pathnames between	cvtname(1)

and SR10 formats	cvtrgy: convert registry between SR9.x	cvtrgy(1)
	bcd: convert to antique media	bcd(6)
dd: convert and	copy a file	dd(1)
	tcop: copy a mag tape	tcop(1)
	cpacl: copy access control list	cpacl(1)
	cp: copy files	cp(1)
rcp: remote file	copy	rcp(1C)
	file cpscr: copy the current display to a	cpscr(1)
uname: UNIX-to-UNIX system	copy uuop, uulog,	uuop(1C)
UNIX-to-UNIX system file	copy uuto, uupick: public	uuto(1)
sum: sum and	count blocks in a file	sum(1)
	wc: word count	wc(1)
hpc: program	counter histogram	hpc(1)
	cp: copy files	cp(1)
	list cpacl: copy access control	cpacl(1)
display to a file	cpscr: copy the current	cpscr(1)
craps: the game of	craps	craps(6)
	craps: the game of craps	craps(6)
modify a device descriptor/	crddf: create, display, or	crddf(1)
stk_font_gen:	create a custom stroke font	stk_font_gen(1)
pad pair and execute a/ dmwin:	create a DM transcript/input	dmwin(1)
	file wbak: create a magnetic media backup	wbak(1)
	crty: create a new type	crty(1)
	node crp: create a process on a remote	crp(1)
for storing/ strfile, unstr:	create a random access file	strfile(6)
	ctags: create a tags file	ctags(1)
window crpad:	create a transcript pad and	crpad(1)
for binding crtyobj:	create a type object module	crtyobj(1)
by messaging C source mkstr:	create an error message file	mkstr(1)
	files admin: create and administer SCCS	admin(1)
device descriptor file crddf:	create, display, or modify a	crddf(1)
database addbib:	create or extend bibliographic	addbib(1)
cribbage: the card game	cribbage	cribbage(6)
cribbage	cribbage: the card game	cribbage(6)
remote node	crp: create a process on a	crp(1)
and window	crpad: create a transcript pad	crpad(1)
filter nroff output for	CRT previewing colrt:	colrt(1)
page: file perusal filter for	CRT viewing more,	more(1)
	crty: create a new type	crty(1)
	crtyobj: create a type object	crtyobj(1)
module for binding	csh: a shell (command	csh(1)
interpreter) with C-like/	(csh only) /a program file,	which(1)
including aliases and paths	ctags: create a tags file	ctags(1)
	tip, cu: connect to a remote system	tip(1C)
	cpscr: copy the	cpscr(1)
set or print identifier of	current display to a file	cpscr(1)
hostname: set or print name of	current host system hostid:	hostid(1)
activity sact: print	current host system	hostname(1)
whoami: print effective	current SCCS file editing	sact(1)
spline: interpolate smooth	current user id	whoami(1)
stk_font_gen: create a	curve	spline(1G)
pre-SR10 to SR10 format	custom stroke font	stk_font_gen(1)
between upper and lowercase/	cvt_font: convert fonts from	cvt_font(1)
between SR9.x and SR10/	cvtname: convert pathnames	cvtname(1)
flake: induce terminal	cvtrgy: convert registry	cvtrgy(1)
directory service whois:	dandruff	flake(6)
	DARPA Internet username	whois(1)

zcat: compress and expand	data compress, uncompress,	compress(1)
display call graph profile	data gprof:	gprof(1)
prof: display profile	data	prof(1)
create or extend bibliographic	database addbib:	addbib(1)
file newaliases: rebuild the	database for the mail aliases	newaliases(1)
join: relational	database operator	join(1)
roffbib: run off bibliographic	database	roffbib(1)
sortbib: sort bibliographic	database	sortbib(1)
date: print and set the	date	date(1)
touch: update	date last modified of a file	touch(1)
	date: print and set the date	date(1)
access control list editor	dbacl: Domain/Dialogue-based	dbacl(1)
	dbx: source code debugger	dbx(1)
	dc: desk calculator	dc(1)
	dd: convert and copy a file	dd(1)
Debugging Environment	dde: Domain Distributed	dde(1)
dbx: source code	debugger	dbx(1)
dde: Domain Distributed	Debugging Environment	dde(1)
dstdump: dump a COFF file's	debugging information to	dstdump(1)
od: octal,	decimal, hex, ASCII dump	od(1)
eqnchar: special character	definitions for eqn	eqnchar(7)
dlty:	delete a type	dlty(1)
file tail:	deliver the last part of a	tail(1)
delta commentary of an SCCS	delta cdc: change the	cdc(1)
delta: make a	delta (change) to an SCCS file	delta(1)
delta cdc: change the	delta commentary of an SCCS	cdc(1)
rmdel: remove a	delta from an SCCS file	rmdel(1)
to an SCCS file	delta: make a delta (change)	delta(1)
comb: combine SCCS	deltas	comb(1)
mesg: permit or	deny messages	mesg(1)
tset: terminal	dependent initialization	tset(1)
tbl, and eqn constructs	deroff: remove nroff, troff,	deroff(1)
whatis:	describe what a command is	whatis(1)
display, or modify a device	descriptor file /create,	crddf(1)
cdmtdesc: edit magtape	descriptor file	cdmtdesc(1)
dc:	desk calculator	dc(1)
lists (ACLs) acl:	details about access control	acl(7)
file:	determine file type	file(1)
create, display, or modify a	device descriptor file crddf:	crddf(1)
lines for finite-width output	device fold: fold long	fold(1)
	df: disk free	df(1)
ratfor: rational FORTRAN	dialect	ratfor(1)
wordy sentences; thesaurus for	diction /explain: print	diction(1)
sentences; thesaurus for/	diction, explain: print wordy	diction(1)
bdiff: big	diff	bdiff(1)
directory comparator	diff: differential file and	diff(1)
file comparison	diff3: three-way differential	diff3(1)
directory comparator diff:	differential file and	diff(1)
diff3: three-way	differential file comparison	diff3(1)
rmdir, rm: remove (unlink)	directories or files	rmdir(1)
remove (unlink) files or	directories rm, rmdir:	rm(1)
cd: change working	directory	cd(1)
diff: differential file and	directory comparator	diff(1)
ls: list contents of	directory	ls(1)
mkdir: make a	directory	mkdir(1)
pwd: working	directory name	pwd(1)

whois: DARPA Internet username	directory service	whois(1)
df:	disk free	df(1)
du: summarize	disk usage	du(1)
messages error: analyze and	disperse compiler error	error(1)
symbol esa:	display address of external	esa(1)
data gprof:	display call graph profile	gprof(1)
snake, snscore:	display chase game	snake(6)
vi: screen oriented (visual)	display editor based on ex	vi(1)
dm/commands -	Display Manager Commands	dm(1)
descriptor/ crddf: create,	display, or modify a device	crddf(1)
graphically dspst:	display process status	dspst(1)
prof:	display profile data	prof(1)
rain: animated raindrops	display	rain(6)
information man:	display reference manual	man(1)
tz: set or	display system time zone	tz(1)
worms: animate worms on a	display terminal	worms(6)
time stamp ts:	display the module name and	ts(1)
was built bldt:	display time operating system	bldt(1)
cpscr: copy the current	display to a file	cpscr(1)
systype:	display version stamp	systype(1)
Environment dde: Domain	Distributed Debugging	dde(1)
rdist: remote file	distribution program	rdist(1)
	dlt: delete a type	dlt(1)
xdmc: execute a	DM command from the shell	xdmc(1)
Display Manager Commands	dm/commands	dm(1)
xownroot: switch between X and	DM ownership of the root while/	xownroot(1)
and execute a/ dmwin: create a	DM transcript/input pad pair	dmwin(1)
Display Manager/ .if '	dm(1) v'2' dm/commands -	dm(1)
Commands .if ' dm(1) v'2'	dm/commands - Display Manager	dm(1)
moire generator	dmoire: Domain/Dialogue-based	dmoire(6)
transcript/input pad pair and/	dmwin: create a DM	dmwin(1)
surface characteristics of a	document style: analyze	style(1)
literature references in	documents /find and insert	refer(1)
Environment dde:	Domain Distributed Debugging	dde(1)
commands and extensions	domain: Domain/OS-specific	domain(1)
games	domain: Domain/OS-specific	domain(6)
Domain/OS features	domain: miscellaneous	domain(7)
control list editor dbacl:	Domain/Dialogue-based access	dbacl(1)
generator dmoire:	Domain/Dialogue-based moire	dmoire(6)
/queue a file for printing by	Domain/OS Aegis print spooler	prf(1)
domain: miscellaneous	Domain/OS features	domain(7)
limits:	Domain/OS limits	limits(7)
and extensions domain:	Domain/OS-specific commands	domain(1)
domain:	Domain/OS-specific games	domain(6)
system server Xapollo: Apollo	Domain/X11 shared X window	Xapollo(1)
graph:	draw a graph	graph(1G)
arithmetic: provide	drill in number facts	arithmetic(6)
graphically	dspst: display process status	dspst(1)
debugging information to	dstdump: dump a COFF file's	dstdump(1)
	du: summarize disk usage	du(1)
emt: emulate a	dumb terminal	emt(1)
information to dstdump:	dump a COFF file's debugging	dstdump(1)
od: octal, decimal, hex, ASCII	dump	od(1)
object file coffdump:	dump selected parts of an	coffdump(1)
echo:	echo arguments	echo(1)
	echo: echo arguments	echo(1)

	ed: text editor	ed(1)
	edfont: edit a character font	edfont(1)
edfont:	edit a character font	edfont(1)
edmdesc:	edit magtape descriptor file	edmdesc(1)
ex,	edit: text editor	ex(1)
sact: print current SCCS file	editing activity	sact(1)
oriented (visual) display	editor based on ex vi: screen	vi(1)
access control list	editor /Domain/Dialogue-based	dbacl(1)
ed: text	editor	ed(1)
ex, edit: text	editor	ex(1)
ld: link	editor for common object files	ld(1)
sed: stream	editor	sed(1)
descriptor file	edmdesc: edit magtape	edmdesc(1)
whoami: print	effective current user id	whoami(1)
for a pattern grep,	egrep, fgrep: search a file	grep(1)
input soelim:	eliminate .so's from nroff	soelim(1)
	emt: emulate a dumb terminal	emt(1)
emt:	emulate a dumb terminal	emt(1)
xterm: terminal	emulator for X	xterm(1)
vt100: VT100 terminal	emulator	vt100
for/ uencode,udecode:	encode/decode a binary file	uencode(1C)
scs: front	end for the SCCS subsystem	scs(1)
convert Arabic numerals to	English number:	number(6)
logger: make	entries in the system log	logger(1)
variables	environ: user environment	environ(7)
Domain Distributed Debugging	Environment dde:	dde(1)
printenv: print out the	environment	printenv(1)
environ: user	environment variables	environ(7)
remove nroff, troff, tbl, and	eqn constructs deroff:	deroff(1)
character definitions for	eqn eqnchar: special	eqnchar(7)
mathematical text	eqn, neqn, checkeq: format	eqn(1)
definitions for eqn	eqnchar: special character	eqnchar(7)
compiler error messages	error: analyze and disperse	error(1)
massaging C/ mkstr: create an	error message file by	mkstr(1)
analyze and disperse compiler	error messages error:	error(1)
spellout: find spelling	errors spell, spellin,	spell(1)
external symbol	esa: display address of	esa(1)
expression expr:	evaluate arguments as an	expr(1)
	ex, edit: text editor	ex(1)
display editor based on	ex /screen oriented (visual)	vi(1)
lpq: spool queue	examination program	lpq(1)
pax: portable archive	exchange	pax(1)
shell xdmc:	execute a DM command from the	xdmc(1)
transcript/input pad pair and	execute a program in it /a DM	dmwin(1)
at:	execute commands later	at(1)
lastcomm: show last commands	executed in reverse order	lastcomm(1)
sleep: suspend	execution for an interval	sleep(1)
UNIX-to-UNIX system command	execution uux:	uux(1)
uncompress, zcat: compress and	expand data compress,	compress(1)
vice versa expand, unexpand:	expand tabs to spaces, and	expand(1)
to spaces, and vice versa	expand, unexpand: expand tabs	expand(1)
sentences; thesaurus/ diction,	explain: print wordy	diction(1)
expression	expr: evaluate arguments as an	expr(1)
expr: evaluate arguments as an	expression	expr(1)
addbib: create or	extend bibliographic database	addbib(1)
commands and	extensions /Domain/OS-specific	domain(1)

esa: display address of	external symbol	esa(1)
programs to implement/ xstr:	extract strings from C	xstr(1)
	f77: FORTRAN 77 compiler	f77(1)
	factor: factoring program	factor(6)
factor:	factoring program	factor(6)
true,	false: provide truth values	true(1)
values	false, true: provide truth	false(1)
fst: print	fault status information	fst(1)
miscellaneous Domain/OS	features domain:	domain(7)
col: filter reverse line	feeds	col(1)
pattern grep, egrep,	fgrep: search a file for a	grep(1)
robots:	fight off villainous robots	robots(6)
diff: differential	file and directory comparator	diff(1)
mkstr: create an error message	file by massaging C source	mkstr(1)
selected parts of an object	file coffdump: dump	coffdump(1)
colrm: remove columns from a	file	colrm(1)
diff3: three-way differential	file comparison	diff(1)
rcp: remote	file copy	rcp(1C)
public UNIX-to-UNIX system	file copy uuto, uupick:	uuto(1)
copy the current display to a	file cpscr:	cpscr(1)
or modify a device descriptor	file crddf: create, display,	crddf(1)
ctags: create a tags	file	ctags(1)
dd: convert and copy a	file	dd(1)
a delta (change) to an SCCS	file delta: make	delta(1)
	file: determine file type	file(1)
rdist: remote	file distribution program	rdist(1)
sact: print current SCCS	file editing activity	sact(1)
edit magtape descriptor	file edmtdesc:	edmtdesc(1)
grep, egrep, fgrep: search a	file for a pattern	grep(1)
Aegis print/ prf: queue a	file for printing by Domain/OS	prf(1)
/unstr: create a random access	file for storing strings	strfile(6)
/cncode/decode a binary	file for transmission via mail	uencode(1C)
fpr: print FORTRAN	file	fpr(1)
get: get a version of an SCCS	file	get(1)
give the first few lines of a	file head:	head(1)
paths/ which: locate a program	file, including aliases and	which(1)
/split a multiroutine FORTRAN	file into individual files	fsplit(1)
split: split a	file into pieces	split(1)
database for the mail aliases	file newaliases: rebuild the	newaliases(1)
viewing more, page:	file perusal filter for CRT	more(1)
pr: print	file	pr(1)
prs: print an SCCS	file	prs(1)
index a magnetic media backup	file rbak: restore or	rbak(1)
rev: reverse lines of a	file	rev(1)
remove a delta from an SCCS	file rmdel:	rmdel(1)
two versions of an SCCS	file sccsdiff: compare	sccsdiff(1)
size: size of an object	file	size(1)
printable strings in an object	file strings: find the	strings(1)
sum: sum and count blocks in a	file	sum(1)
hier:	file system hierarchy	hier(7)
deliver the last part of a	file tail:	tail(1)
update date last modified of a	file touch:	touch(1)
ftp: ARPANET	file transfer program	ftp(1C)
tftp: trivial	file transfer protocol	tftp(1C)
file: determine	file type	file(1)
undo a previous get of an SCCS	file unget:	unget(1)

report repeated lines in a	file uniq:	uniq(1)
val: validate SCCS	file	val(1)
create a magnetic media backup	file wbak:	wbak(1)
were used to construct a	file /of object modules	what(1)
basename: strip	filename affixes	basename(1)
create and administer SCCS	files admin:	admin(1)
/ccat: compress and uncompress	files, and then cat them	compact(1)
checknr: check nroff/troff	files	checknr(1)
cmp: compare two	files	cmp(1)
lines common to two sorted	files comm: select or reject	comm(1)
cp: copy	files	cp(1)
to dstdump: dump a COFF	file's debugging information	dstdump(1)
find: find	files	find(1)
FORTRAN file into individual	files /split a multiroutine	fsplit(1)
link editor for common object	files ld:	ld(1)
maintain (make) related	files mk, mkconv, memname:	mk(1)
mv: move or rename	files	mv(1)
lpr: print	files offline	lpr(1)
rm, rmdir: remove (unlink)	files or directories	rm(1)
remove (unlink) directories or	files rmdir, rm:	rmdir(1)
sort: sort or merge	files	sort(1)
iso: convert	files to ISO format	iso(1)
more, page: file perusal	filter for CRT viewing	more(1)
previewing colctr:	filter nroff output for CRT	colctr(1)
col:	filter reverse line feeds	col(1)
plot: graphics	filters	plot(1G)
references in/ refer:	find and insert literature	refer(1)
find:	find files	find(1)
	find: find files	find(1)
look:	find lines in a sorted list	look(1)
object library lorder:	find ordering relation for an	lorder(1)
spell, spellin, spellout:	find spelling errors	spell(1)
an object file strings:	find the printable strings in	strings(1)
lookup program	finger: user information	finger(1)
fold: fold long lines for	finite-width output device	fold(1)
	fish: play lqGo Fishrq	fish(6)
fish: play lqGo	Fishrq	fish(6)
tee: pipe	fitting	tee(1)
dandruff	flake: induce terminal	flake(6)
	fnt: simple text formatter	fnt(1)
finite-width output device	fold: fold long lines for	fold(1)
finite-width output/ fold:	fold long lines for	fold(1)
bdf2snf - BDF to SNF	font compiler for X11	bdf2snf(1)
edfont: edit a character	font	edfont(1)
create a custom stroke	font stk_font_gen:	stk_font_gen(1)
characters within a	font tr_font: transliterate	tr_font(1)
format cvt_font: convert	fonts from pre-SR10 to SR10	cvt_font(1)
rwmt: read/write	foreign magtapes	rwmt(1)
indent: indent and	format C program source	indent(1)
fonts from pre-SR10 to SR10	format cvt_font: convert	cvt_font(1)
iso: convert files to ISO	format	iso(1)
eqn, neqn, checkeq:	format mathematical text	eqn(1)
OBJ format modules to COFF	format modules /convert	obj2coff(1)
modules obj2coff: convert OBJ	format modules to COFF format	obj2coff(1)
troff tbl:	format tables for nroff or	tbl(1)
between SR9.x and SR10	formats /convert registry	cvtrgy(1)

fmt: simple text	formatter	fmt(1)
troff: text	formatting and typesetting	troff(1)
ms: text	formatting macros	ms(7)
nroff: text	formatting	nroff(1)
me: macros for	formatting papers	me(7)
f77:	FORTRAN 77 compiler	f77(1)
ratfor: rational	FORTRAN dialect	ratfor(1)
fpr: print	FORTRAN file	fpr(1)
fsplit: split a multiroutine	FORTRAN file into individual/	fsplit(1)
struct: structure	FORTRAN programs	struct(1)
hopefully interesting, adage	fortune: print a random,	fortune(6)
	fpr: print FORTRAN file	fpr(1)
df: disk	free	df(1)
colrm: remove columns	from a file	colrm(1)
rmde1: remove a delta	from an SCCS file	rmde1(1)
if mail arrives and who it is	from biff: be notified	biff(1)
shared/ xstr: extract strings	from C programs to implement	xstr(1)
from: who is my mail	from?	from(1)
soelim: eliminate .so's	from nroff input	soelim(1)
cvt_font: convert fonts	from pre-SR10 to SR10 format	cvt_font(1)
queue lprm: remove jobs	from the line-printer spooling	lprm(1)
xdmc: execute a DM command	from the shell	xdmc(1)
	from: who is my mail from?	from(1)
subsystem sccs:	front end for the SCCS	sccs(1)
FORTRAN file into individual/	fsplit: split a multiroutine	fsplit(1)
information	fst: print fault status	fst(1)
program	ftp: ARPANET file transfer	ftp(1C)
a tropical adventure	game battlestar:	battlestar(6)
cfscorcs: the solitaire card	game canfield canfield,	canfield(6)
cribbage: the card	game cribbage	cribbage(6)
Computer version of the	game hangman hangman:	hangman(6)
a multi-player multi-terminal	game hunt:	hunt(6)
Mastermind guessing	game mastermind:	mastermind(6)
monop: Monopoly	game	monop(6)
moo: guessing	game	moo(6)
backgammon: the	game of backgammon	backgammon(6)
btlgammon: the	game of backgammon	btlgammon(6)
teachgammon: teach the	game of backgammon	teachgammon(6)
bj: the	game of blackjack	bj(6)
boggle: play the	game of boggle	boggle(6)
craps: the	game of craps	craps(6)
wump: the	game of hunt-the-wumpus	wump(6)
puzzle: puzzle	game	puzzle(6)
snake, snsore: display chase	game	snake(6)
trek: trekkie	game	trek(6)
worm: play the growing worm	game	worm(6)
domain: Domain/OS-specific	games	domain(6)
intro: introduction to	games	intro(6)
maze:	generate a maze	maze(6)
pattern lptest:	generate lineprinter ripple	lptest(1)
Domain/Dialoguc-based moire	generator dmoire:	dmoire(6)
programs lex:	generator of lexical analysis	lex(1)
random: random number	generator	random(6)
get:	get a version of an SCCS file	get(1)
file	get: get a version of an SCCS	get(1)
unget: undo a previous	get of an SCCS file	unget(1)

for prism prism:	get processor type truth value	prism(1)
tty:	get terminal name	tty(1)
profile data	gprof: display call graph	gprof(1)
	graph: draw a graph	graph(1G)
graph: draw a	graph	graph(1G)
gprof: display call	graph profile data	gprof(1)
dspst: display process status	graphically	dspst(1)
	graphics filters	plot(1G)
plot:	grep, egrep, fgrep: search a	grep(1)
file for a pattern	grind nice listings of	vgrind(1)
programs vgrind:	group	chgrp(1)
chgrp: change	group memberships	groups(1)
groups: show	groups of programs /update	make(1)
and regenerate programs or	groups: show group memberships	groups(1)
	grow vines	vine(6)
vine:	growing worm game	worm(6)
worm: play the	guess the word	bthangman(6)
btlangman:	guessing game	mastermind(6)
mastermind:	guessing game	moo(6)
moo:	handle remote mail received	rmail(1)
via uucp rmail:	hangman: Computer version of	hangman(6)
the game hangman	hangman hangman:	hangman(6)
Computer version of the game	help: ask for SCCS help	help(1)
	help	help(1)
help: ask for SCCS	hex, ASCII dump	od(1)
od: octal, decimal,	hier: file system hierarchy	hier(7)
	hierarchy	hier(7)
hier: file system	histogram	hpc(1)
hpc: program counter	hopefully interesting, adage	fortune(6)
fortune: print a random,	host status of local machines	ruptime(1C)
ruptime: show	host system hostid: set	hostid(1)
or print identifier of current	host system hostname:	hostname(1)
set or print name of current	hostid: set or print	hostid(1)
identifier of current host/	hostname: set or print name of	hostname(1)
current host system	hpc: program counter histogram	hpc(1)
	hunt: a multi-player	hunt(6)
multi-terminal game	hunt-the-wumpus	wump(6)
wump: the game of	ID temporarily	su(1)
su: substitute user	id whoami:	whoami(1)
print effective current user	identifier of current host	hostid(1)
system hostid: set or print	.if ' . dm(1) v'2' dm/commands	dm(1)
Display Manager Commands	ifndef'd lines	unifdef(1)
unifdef: remove	/strings from C programs to	xstr(1)
/strings from C programs to	implement shared strings	which(1)
which: locate a program file,	including aliases and paths/	indent(1)
source indent:	indent and format C program	indent(1)
program source	indent: indent and format C	indent(1)
file rbak: restore or	index a magnetic media backup	rbak(1)
lookbib: build inverted	index for a indxbib,	lookbib(1)
index	index	ptx(1)
and teletypes last:	indicate last logins of users	last(1)
flake:	induce terminal dandruff	flake(6)
inverted index for a	indxbib, lookbib: build	lookbib(1)
tset: terminal dependent	initialization	tset(1)
library	inlib: install a user-supplied	inlib(1)
eliminate .so's from nroff	input soelim:	soelim(1)
uustat: uucp status	inquiry and job control	uustat(1)

in documents refer: find and	insert literature references	refer(1)
inty:	install a new type	inty(1)
intm	install a type manager	intm(1)
library inlib:	install a user-supplied	inlib(1)
install:	install binaries	install(1)
	install: install binaries	install(1)
llib: list	installed libraries	llib(1)
lty: list	installed types	lty(1)
print a random, hopefully	interesting, adage fortune:	fortune(6)
bgcolor: make	interesting background colors	bgcolor(6)
protocol telnet: user	interface to the TELNET	telnet(1C)
service whois: DARPA	Internet username directory	whois(1)
spline:	interpolate smooth curve	spline(1G)
syntax csh: a shell (command	interpreter) with C-like	csh(1)
suspend execution for an	interval sleep:	sleep(1)
	intm install a type manager	intm(1)
commands	intro: introduction to	intro(1)
	intro: introduction to games	intro(6)
	intro: introduction to commands	intro(1)
	intro: introduction to games	intro(6)
	inty: install a new type	inty(1)
indxib, lookbib: build	inverted index for a	lookbib(1)
multi-user wooden ships and	iron men sail:	sail(6)
format	iso: convert files to ISO	iso(1)
iso: convert files to	ISO format	iso(1)
operator	join: relational database	join(1)
msgs: system messages and	junk mail program	msgs(1)
characteristics	kbm: set/display keyboard	kbm(1)
kbm: set/display	keyboard characteristics	kbm(1)
apropos: locate commands by	keyword lookup	apropos(1)
process	kill: terminate or signal a	kill(1)
quiz: test your	knowledge	quiz(6)
language ksh: the	Korn shell command programming	ksh(1)
programming language	ksh: the Korn shell command	ksh(1)
scanning and processing	language awk: pattern	awk(1)
arbitrary-precision arithmetic	language bc:	bc(1)
Korn shell command programming	language ksh: the	ksh(1)
scanning and processing	language nawk: pattern	nawk(1)
sh: the Bourne shell command	language	sh(1)
the address space	las: list objects mapped into	las(1)
executed in reverse order	lastcomm: show last commands	lastcomm(1)
libraries lbr2ar: convert	lbr libraries to SR10 archive	lbr2ar(1)
to SR10 archive libraries	lbr2ar: convert lbr libraries	lbr2ar(1)
	lcm: load a color map	lcm(1)
object files	ld: link editor for common	ld(1)
remind you when you have to	leave leave:	leave(1)
have to leave	leave: remind you when you	leave(1)
analysis programs	lex: generator of lexical	lex(1)
lex: generator of	lexical analysis programs	lex(1)
lbr libraries to SR10 archive	libraries lbr2ar: convert	lbr2ar(1)
llib: list installed	libraries	llib(1)
convert archives to random	libraries ranlib:	ranlib(1)
libraries lbr2ar: convert lbr	libraries to SR10 archive	lbr2ar(1)
inlib: install a user-supplied	library	inlib(1)
relation for an object	library lorder: find ordering	lorder(1)
ar: archive and	library maintainer	ar(1)

	limits: Domain/OS limits	limits(7)
limits: Domain/OS	limits	limits(7)
col: filter reverse	line feeds	col(1)
print: pr to the	line printer	print(1)
lptest: generate	lineprinter ripple pattern	lptest(1)
lprm: remove jobs from the	line-printer spooling queue	lprm(1)
files comm: select or reject	lines common to two sorted	comm(1)
device fold: fold long	lines for finite-width output	fold(1)
uniq: report repeated	lines in a file	uniq(1)
look: find	lines in a sorted list	look(1)
head: give the first few	lines of a file	head(1)
rev: reverse	lines of a file	rev(1)
unifdef: remove ifdef'ed	lines	unifdef(1)
files ld:	link editor for common object	ld(1)
ln: make	links	ln(1)
	lint: a C program verifier	lint(1)
lsacl:	list access control list	lsacl(1)
chacl: change access control	list	chacl(1)
ls:	list contents of directory	ls(1)
cpacl: copy access control	list	cpacl(1)
/access control	list editor	dbacl(1)
llib:	list installed libraries	llib(1)
lty:	list installed types	lty(1)
llkob:	list locked objects	llkob(1)
look: find lines in a sorted	list	look(1)
lsacl: list access control	list	lsacl(1)
nm: print name	list	nm(1)
address space las:	list objects mapped into the	las(1)
node users: compact	list of users who are on a	users(1)
vgrind: grind nice	listings of programs	vgrind(1)
details about access control	lists (ACLs) acl:	acl(7)
refer: find and insert	literature references in/	refer(1)
	llib: list installed libraries	llib(1)
	llkob: list locked objects	llkob(1)
	ln: make links	ln(1)
lcm:	load a color map	lcm(1)
including aliases and/ which:	locate a program file,	which(1)
lookup apropos:	locate commands by keyword	apropos(1)
manual for program whereis:	locate source, binary, and/or	whereis(1)
llkob: list	locked objects	llkob(1)
make entries in the system	log logger:	logger(1)
rwho: who's	logged in on local machines	rwho(1C)
system log	logger: make entries in the	logger(1)
start_sh: start a	log-in Bourne shell	start_sh(1)
start_csh: start a	log-in C shell	start_csh(1)
rlogin: remote	login	rlogin(1C)
	login: sign on	login(1)
last: indicate last	logins of users and teletypes	last(1)
output device fold: fold	long lines for finite-width	fold(1)
uptime: show how	long system has been up	uptime(1)
for a indxbib,	lookbib: build inverted index	lookbib(1)
locate commands by keyword	lookup apropos:	apropos(1)
finger: user information	lookup program	finger(1)
for an object library	lorder: find ordering relation	lorder(1)
/pathnames between upper and	lowercase and preserve colons	cvname(1)
program	lpq: spool queue examination	lpq(1)

line-printer spooling queue	lpr: print files offline	lpr(1)
ripple pattern	lprm: remove jobs from the	lprm(1)
fish: play	lptest: generate lineprinter	lptest(1)
melt:	lqGo Fishrq	fish(6)
	lqmeltrq the screen	melt(6)
list	ls: list contents of directory	ls(1)
	lsacl: list access control	lsacl(1)
	lty: list installed types	lty(1)
show host status of local	m4: macro processor	m4(1)
rwwho: who's logged in on local	machines ruptime:	ruptime(1C)
	machines	rwwho(1C)
	m4: macro processor	m4(1)
	me: macros for formatting papers	me(7)
ms: text formatting	macros	ms(7)
man:	macros to typeset manual	man(7)
tcopy: copy a	mag tape	tcopy(1)
rbak: restore or index a	magnetic media backup file	rbak(1)
wbak: create a	magnetic media backup file	wbak(1)
program mt:	magnetic tape manipulating	mt(1)
edmtdesc: edit	magtape descriptor file	edmtdesc(1)
rwmt: read/write foreign	magtapes	rwmt(1)
rebuild the database for the	mail aliases file newaliases:	newaliases(1)
binmail: send or receive	mail among users	binmail(1)
from biff: be notified if	mail arrives and who it is	biff(1)
from: who is my	mail from?	from(1)
mail: send and receive	mail	mail(1)
msgs: system messages and junk	mail program	msgs(1)
rmail: handle remote	mail received via uucp	rmail(1)
	mail: send and receive mail	mail(1)
file for transmission via	mail /encode/decode a binary	uencode(1C)
mk, mkconv, membername:	maintain (make) related files	mk(1)
ar: archive and library	maintainer	ar(1)
SCCS file delta:	make a delta (change) to an	delta(1)
mkdir:	make a directory	mkdir(1)
mkapr:	make an Apollo product report	mkapr(1)
logger:	make entries in the system log	logger(1)
colors bgcolor:	make interesting background	bgcolor(6)
ln:	make links	ln(1)
mkconv, membername: maintain	(make) related files mk,	mk(1)
session script:	make typescript of terminal	script(1)
programs or groups of/	make: update and regenerate	make(1)
information	man: display reference manual	man(1)
	man: macros to typeset manual	man(7)
dm/commands - Display	Manager Commands	dm(1)
uwm - a window	manager for X	uwm(1)
intm install a type	manager	intm(1)
mt: magnetic tape	manipulating program	mt(1)
locate source, binary, and/or	manual for program whereis:	whereis(1)
man: display reference	manual information	man(1)
man: macros to typeset	manual	man(7)
lcm: load a color	map	lcm(1)
ascii:	map of ASCII character set	ascii(7)
las: list objects	mapped into the address space	las(1)
an error message file by	massaging C source /create	mkstr(1)
mastermind:	Mastermind guessing game	mastermind(6)
guessing game	mastermind: Mastermind	mastermind(6)

eqn, neqn, checkeq: format	mathematical text	eqn(1)
maze: generate a	maze: generate a maze	maze(6)
maze: generate a	maze	maze(6)
restore or index a magnetic	media backup file rbak:	rbak(1)
wbak: create a magnetic	media backup file	wbak(1)
bcd: convert to antique	media	bcd(6)
	melt: lqmeltrq the screen	melt(6)
related files mk, mkconv,	membername: maintain (make)	mk(1)
groups: show group	memberships	groups(1)
sort: sort or	merge files	sort(1)
source mkstr: create an error	msg: permit or deny messages	msg(1)
status code value to text	message file by massaging C	mkstr(1)
msgs: system	message stcode: translate	stcode(1)
and disperse compiler error	messages and junk mail program	msgs(1)
mcsg: permit or deny	messages error: analyze	error(1)
mille: play	messages	msg(1)
	Mille Bournes	mille(6)
	mille: play Mille Bournes	mille(6)
maintain (make) related files	mk, mkconv, membername:	mk(1)
report	mkapr: make an Apollo product	mkapr(1)
(make) related files mk,	mkconv, membername: maintain	mk(1)
	mkdir: make a directory	mkdir(1)
file by massaging C source	mkstr: create an error message	mkstr(1)
chmod: change	mode	chmod(1)
touch: update date last	modified of a file	touch(1)
crddf: create, display, or	modify a device descriptor/	crddf(1)
crtobj: create a type object	module for binding	crtobj(1)
ts: display the	module name and time stamp	ts(1)
format modules to COFF format	modules obj2coff: convert OBJ	obj2coff(1)
obj2coff: convert OBJ format	modules to COFF format modules	obj2coff(1)
/show what versions of object	modules were used to construct/	what(1)
dmoire: Domain/Dialogue-based	moire generator	dmoire(6)
	monop: Monopoly game	monop(6)
	monop: Monopoly game	monop(6)
	moo: guessing game	moo(6)
filter for CRT viewing	more, page: file perusal	more(1)
tpm: set/display touchpad and	mouse characteristics	tpm(1)
mv:	move or rename files	mv(1)
	ms: text formatting macros	ms(7)
mail program	msgs: system messages and junk	msgs(1)
program	mt: magnetic tape manipulating	mt(1)
game hunt: a	multi-player multi-terminal	hunt(6)
individual/ fsplit: split a	multiroutine FORTRAN file into	fsplit(1)
hunt: a multi-player	multi-terminal game	hunt(6)
iron men sail:	multi-user wooden ships and	sail(6)
	mv: move or rename files	mv(1)
processing language	nawk: pattern scanning and	nawk(1)
mathematical text eqn,	neqn, checkeq: format	eqn(1)
nS: show	network status	netstat(1)
system X - a portable,	network transparent window	X(1)
database for the mail aliases/	newaliases: rebuild the	newaliases(1)
vgrind: grind	nice listings of programs	vgrind(1)
a changed priority (sh only)	nice, nohup: run a command at	nice(1)
	nm: print name list	nm(1)
create a process on a remote	node crp:	crp(1)
list of users who are on a	node users: compact	users(1)

changed priority (sh/ nice, who it is from biff: be soelim: eliminate .so's from tbl: format tables for previewing colcrt: filter	nohup: run a command at a notified if mail arrives and nroff input nroff or troff nroff output for CRT nroff: text formatting nroff, troff, tbl, and eqn nroff/troff files nS: show network status numerals to English OBJ format modules to COFF obj2coff: convert OBJ format object file coffdump: object file object file strings: find object files object library loader: object module for binding object modules were used to/ objects objects mapped into the octal, decimal, hex, ASCII od: octal, decimal, hex, ASCII offline only) /nohup: run a command only) /program file, including operating system was built operator options ordering relation for an oriented (visual) display output device fold: fold output for CRT previewing ownership of the root while pad and window pad pair and execute a program/ page: file perusal filter for page size pages /miscellaneous pagesize: print system page pair and execute a program in papers pathnames between upper and paths (csh only) /a program pattern grep, egrep, pattern lptest: pattern scanning and pattern scanning and pax: portable archive exchange permit or deny messages ptx: permuted index perusal filter for CRT viewing pieces pipe fitting play lqGo Fishrq play Mille Bournes	nice(1) biff(1) soelim(1) tbl(1) colcrt(1) nroff(1) deroff(1) checknr(1) netstat(1) number(6) obj2coff(1) obj2coff(1) coffdump(1) size(1) strings(1) ld(1) lorder(1) crtobj(1) what(1) llkob(1) las(1) od(1) od(1) lpr(1) nice(1) which(1) bld(1) join(1) sty(1) lorder(1) vi(1) fold(1) colcrt(1) xownroot(1) crpad(1) dmwin(1) more(1) pagesize(1) intro(7) pagesize(1) dmwin(1) mc(7) cvtname(1) which(1) grep(1) lptest(1) awk(1) nawk(1) pax(1) msg(1) ptx(1) more(1) split(1) tee(1) fish(6) mille(6)
---	---	--

boggle:	play the game of boggle	boggle(6)
worm:	play the growing worm game	worm(6)
	plot: graphics filters	plot(1G)
pax:	portable archive exchange	pax(1)
window system X - a	portable, network transparent	X(1)
	pr: print file	pr(1)
print:	pr to the line printer	print(1)
xset - user	preference utility for X	xset(1)
upper and lowercase and	preserve colons /between	cvtname(1)
cvt_font: convert fonts from	pre-SR10 to SR10 format	cvt_font(1)
filter nroff output for CRT	previewing colcrt:	colcrt(1)
unget: undo a	previous get of an SCCS file	unget(1)
by Domain/OS Aegis print/	prf: queue a file for printing	prf(1)
primes: print	prime numbers	primes(6)
	primes: print prime numbers	primes(6)
blfortune:	print a random comment	blfortune(6)
interesting, adage fortune:	print a random, hopefully	fortune(6)
	prs: print an SCCS file	prs(1)
date:	print and set the date	date(1)
cal:	print calendar	cal(1)
cat: catenate and	print	cat(1)
editing activity sact:	print current SCCS file	sact(1)
id whoami:	print effective current user	whoami(1)
	fst: print fault status information	fst(1)
	pr: print file	pr(1)
	lpr: print files offline	lpr(1)
	fpr: print FORTRAN file	fpr(1)
host system hostid: set or	print identifier of current	hostid(1)
banner:	print large banner on printer	banner(6)
nm:	print name list	nm(1)
system hostname: set or	print name of current host	hostname(1)
printenv:	print out the environment	printenv(1)
	print: pr to the line printer	print(1)
primes:	print prime numbers	primes(6)
tb:	print process traceback	tb(1)
printing by Domain/OS Aegis	print spooler /a file for	prf(1)
pagesize:	print system page size	pagesize(1)
waiting to be run atq:	print the queue of jobs	atq(1)
thesaurus/ diction, explain:	print wordy sentences;	diction(1)
file strings: find the	printable strings in an object	strings(1)
environment	printenv: print out the	printenv(1)
banner: print large banner on	printer	banner(6)
print: pr to the line	printer	print(1)
print/ prf: queue a file for	printing by Domain/OS Aegis	prf(1)
run a command at a changed	priority (sh only) /nohup:	nice(1)
truth value for prism	prism: get processor type	prism(1)
processor type truth value for	prism prism: get	prism(1)
kill: terminate or signal a	process	kill(1)
crp: create a	process on a remote node	crp(1)
dspst: display	process status graphically	dspst(1)
ps:	process status	ps(1)
tb: print	process traceback	tb(1)
wait: await completion of	process	wait(1)
awk: pattern scanning and	processing language	awk(1)
nawk: pattern scanning and	processing language	nawk(1)
m4: macro	processor	m4(1)

prism prism: get	processor type truth value for	prism(1)
mkapr: make an Apollo	product report	mkapr(1)
gprof: display call graph	prof: display profile data	prof(1)
prof: display	profile data	gprof(1)
ksh: the Korn shell command	profile data	prof(1)
user interface to the TELNET	programming language	ksh(1)
tftp: trivial file transfer	protocol telnet:	telnet(1C)
arithmetic:	protocol	tftp(1C)
false, true:	provide drill in number facts	arithmetic(6)
true, false:	provide truth values	false(1)
	provide truth values	true(1)
	prs: print an SCCS file	prs(1)
	ps: process status	ps(1)
	ptx: permuted index	ptx(1)
puzzle:	puzzle game	puzzle(6)
	puzzle: puzzle game	puzzle(6)
your screen into a scramble	puzzle scramble: turn	scramble(6)
Domain/OS Aegis print/ prf:	pwd: working directory name	pwd(1)
lpq: spool	queue a file for printing by	prf(1)
from the line-printer spooling	queue examination program	lpq(1)
run atq: print the	queue lprm: remove jobs	lprm(1)
	queue of jobs waiting to be	atq(1)
display	quiz: test your knowledge	quiz(6)
rain: animated	rain: animated raindrops	rain(6)
strfile, unstr: create a	raindrops display	rain(6)
bitfortune: print a	random access file for storing/	strfile(6)
adage fortune: print a	random comment	bitfortune(6)
ranlib: convert archives to	random, hopefully interesting,	fortune(6)
random:	random libraries	ranlib(1)
generator	random number generator	random(6)
random libraries	random: random number	random(6)
dialect	ranlib: convert archives to	ranlib(1)
ratfor:	ratfor: rational FORTRAN	ratfor(1)
magnetic media backup file	rational FORTRAN dialect	ratfor(1)
	rbak: restore or index a	rbak(1)
distribution program	rcp: remote file copy	rcp(1C)
rwmt:	rdist: remote file	rdist(1)
swapul:	read/write foreign magtapes	rwmt(1)
mail aliases file newaliases:	rearrange underlining	swapul(1)
binmail: send or	rebuild the database for the	newaliases(1)
mail: send and	receive mail among users	binmail(1)
rmail: handle remote mail	receive mail	mail(1)
literature references in/	received via uuwp	rmail(1)
man: display	refer: find and insert	refer(1)
/find and insert literature	reference manual information	man(1)
of programs make: update and	references in documents	refer(1)
SR10 formats cvtrgy: convert	regenerate programs or groups	make(1)
sorted files comm: select or	registry between SR9.x and	cvtrgy(1)
membername: maintain (make)	reject lines common to two	comm(1)
lorder: find ordering	related files mk, mkconv,	mk(1)
join:	relation for an object library	lorder(1)
strip: remove symbols and	relational database operator	join(1)
leave leave:	relocation bits	strip(1)
calendar:	remind you when you have to	leave(1)
rcp:	reminder service	calendar(1)
	remote file copy	rcp(1C)

program rdist:	remote file distribution	rdist(1)
rlogin:	remote login	rlogin(1C)
rmail: handle	remote mail received via uucp	rmail(1)
crp: create a process on a	remote node	crp(1)
rS:	remote shell	rsh(1C)
tip, cu: connect to a	remote system	tip(1C)
file rmdel:	remove a delta from an SCCS	rmdel(1)
colrm:	remove columns from a file	colrm(1)
unifdef:	remove ifdef'd lines	unifdef(1)
line-printer spooling/ lprm:	remove jobs from the	lprm(1)
atrm:	remove jobs spooled by	atrm(1)
eqn constructs derooff:	remove nroff, troff, tbl, and	deroff(1)
bits strip:	remove symbols and relocation	strip(1)
files rmdir, rm:	remove (unlink) directories or	rmdir(1)
directories rm, rmdir:	remove (unlink) files or	rm(1)
mv: move or	rename files	mv(1)
uniq: report	repeated lines in a file	uniq(1)
yes: be	repetitively affirmative	yes(1)
mkapr: make an Apollo product	report	mkapr(1)
file uniq:	report repeated lines in a	uniq(1)
media backup file rbak:	restore or index a magnetic	rbak(1)
rev:	reverse lines of a file	rev(1)
col: filter	reverse line feeds	col(1)
rev:	reverse lines of a file	rev(1)
show last commands executed in	reverse order lastcomm:	lastcomm(1)
revscr:	reverse screen	revscr(6)
revscr:	reverse screen	revscr(6)
lptest: generate lineprinter	ripple pattern	lptest(1)
rlogin:	remote login	rlogin(1C)
rm: remove (unlink)	rmdir(1)	rmdir(1)
rm, rmdir: remove (unlink)	rm(1)	rm(1)
rmail: handle remote mail	rmail(1)	rmail(1)
rmdel: remove a delta from an	rmdel(1)	rmdel(1)
rmdir: remove (unlink) files	rm(1)	rm(1)
rmdir, rm: remove (unlink)	rmdir(1)	rmdir(1)
robots: fight off villainous	robots(6)	robots(6)
robots: fight off villainous	robots(6)	robots(6)
roffbib: run off bibliographic	roffbib(1)	roffbib(1)
root while the X server is/	xownroot(1)	xownroot(1)
rS: remote shell	rsh(1C)	rsh(1C)
priority (sh/ nice, nohup:	run a command at a changed	nice(1)
queue of jobs waiting to be	run atq: print the	atq(1)
roffbib:	run off bibliographic database	roffbib(1)
the root while the X server is	running /X and DM ownership of	xownroot(1)
local machines	runtime: show host status of	runtime(1C)
machines	rwho: who's logged in on local	rwho(1C)
magtapes	rwmt: read/write foreign	rwmt(1)
editing activity	sact: print current SCCS file	sact(1)
and iron men	sail: multi-user wooden ships	sail(6)
language awk: pattern	scanning and processing	awk(1)
language nawk: pattern	scanning and processing	nawk(1)
the delta commentary of an	SCCS delta cdc: change	cdc(1)
comb: combine	SCCS deltas	comb(1)
make a delta (change) to an	SCCS file delta:	delta(1)
sact: print current	SCCS file editing activity	sact(1)
get: get a version of an	SCCS file	get(1)

prs: print an	SCCS file	prs(1)
rm: remove a delta from an	SCCS file	rm(1)
compare two versions of an	SCCS file	scsdiff(1)
undo a previous get of an	SCCS file	unget(1)
val: validate	SCCS file	val(1)
admin: create and administer	SCCS files	admin(1)
subsystem	scs: front end for the SCCS	scs(1)
help: ask for	SCCS help	help(1)
scs: front end for the	SCCS subsystem	scs(1)
of an SCCS file	scsdiff: compare two versions	scsdiff(1)
turn your screen into a	scramble puzzle	scramble(6)
into a scramble puzzle	scramble: turn your screen	scramble(6)
	scrattr: screen attributes	scrattr(1)
	scrattr: screen attributes	scrattr(1)
clear: clear terminal	screen	clear(1)
scramble: turn your	screen into a scramble puzzle	scramble(6)
melt: lqmeltrq the	screen	melt(6)
display editor based on/ vi:	screen oriented (visual)	vi(1)
revscr: reverse	screen	revscr(6)
scrt: set/show	screen timeout	scrt(1)
terminal session	script: make typescript of	script(1)
	scrt: set/show screen timeout	scrt(1)
grep, egrep, fgrep:	search a file for a pattern	grep(1)
	sed: stream editor	sed(1)
to two sorted files comm:	select or reject lines common	comm(1)
file coffdump: dump	selected parts of an object	coffdump(1)
mail:	send and receive mail	mail(1)
users binmail:	send or receive mail among	binmail(1)
diction, explain: print wordy	sentences; thesaurus for/	diction(1)
of the root while the X	server is running /ownership	xownroot(1)
shared X window system	server /Apollo Domain/X11	Xapollo(1)
X - X Window System	server	Xserver(1)
make typescript of terminal	session script:	script(1)
characteristics kbm:	set/display keyboard	kbm(1)
characteristics tpm:	set/display touchpad and mouse	tpm(1)
settings vsize:	set/display VT100 window	vsize(1)
scrt:	set/show screen timeout	scrt(1)
set/display VT100 window	settings vsize:	vsize(1)
command at a changed priority	(sh only) nice, nohup: run a	nice(1)
language	sh: the Bourne shell command	sh(1)
from C programs to implement	shared strings /strings	xstr(1)
Xapollo: Apollo Domain/X11	shared X window system/	Xapollo(1)
with C-like syntax csh: a	shell (command interpreter)	csh(1)
sh: the Bourne	shell command language	sh(1)
language ksh: the Korn	shell command programming	ksh(1)
ver: change the version of	shell commands	ver(1)
rS: remote	shell	rsh(1C)
start_csh: start a log-in C	shell	start_csh(1)
start a log-in Bourne	shell start_sh:	start_sh(1)
execute a DM command from the	shell xdm:	xdm(1)
sail: multi-user wooden	ships and iron men	sail(6)
login:	sign on	login(1)
kill: terminate or	signal a process	kill(1)
fmt:	simple text formatter	fmt(1)
size:	size of an object file	size(1)
pagesize: print system page	size	pagesize(1)

	size: size of an object file	size(1)
an interval	sleep: suspend execution for	sleep(1)
spline: interpolate	smooth curve	spline(1G)
game	snake, snscore: display chase	snake(6)
bdf2osnf - BDF to	SNF font compiler for X11	bdf2osnf(1)
snake,	nscore: display chase game	snake(6)
nroff input	soelim: eliminate .so's from	soelim(1)
canfield, cfscores: the	solitaire card game canfield	canfield(6)
sortbib:	sort bibliographic database	sortbib(1)
sort:	sort or merge files	sort(1)
	sort: sort or merge files	sort(1)
tsort: topological	sort	tsort(1)
database	sortbib: sort bibliographic	sortbib(1)
or reject lines common to two	sorted files comm: select	comm(1)
look: find lines in a	sorted list	look(1)
soelim: eliminate	.so's from nroff input	soelim(1)
for program whereis: locate	source, binary, and/or manual	whereis(1)
dbx:	source code debugger	dbx(1)
indent and format C program	source indent:	indent(1)
message file by massaging C	source mkstr: create an error	mkstr(1)
mapped into the address	space las: list objects	las(1)
/unexpand: expand tabs to	spaces, and vice versa	expand(1)
spelling errors	spell, spellin, spellout: find	spell(1)
spelling errors spell,	spellin, spellout: find	spell(1)
spell, spellin, spellout: find	spelling errors	spell(1)
spell, spellin,	spellout: find spelling errors	spell(1)
curve	spline: interpolate smooth	spline(1G)
split:	split a file into pieces	split(1)
file into individual/ fsplit:	split a multiroutine FORTRAN	fsplit(1)
pieces	split: split a file into	split(1)
program lpq:	spool queue examination	lpq(1)
atrm: remove jobs	spooled by	atrm(1)
by Domain/OS Aegis print	spooler /a file for printing	prf(1)
jobs from the line-printer	spooling queue lprm: remove	lprm(1)
/convert lbr libraries to	SR10 archive libraries	lbr2ar(1)
convert fonts from pre-SR10 to	SR10 format cvt_font:	cvt_font(1)
registry between SR9.x and	SR10 formats cvtrgy: convert	cvtrgy(1)
/convert registry between	SR9.x and SR10 formats	cvtrgy(1)
systype: display version	stamp	systype(1)
the module name and time	stamp ts: display	ts(1)
start_sh:	start a log-in Bourne shell	start_sh(1)
start_csh:	start a log-in C shell	start_csh(1)
shell	start_csh: start a log-in C	start_csh(1)
Bourne shell	start_sh: start a log-in	start_sh(1)
message stcode: translate	status code value to text	stcode(1)
dsps: display process	status graphically	dsps(1)
fst: print fault	status information	fst(1)
uustat: uucp	status inquiry and job control	uustat(1)
status	status	netstat(1)
nS: show network	status of local machines	ruptime(1C)
ruptime: show host	status	ps(1)
ps: process	stcode: translate status code	stcode(1)
value to text message	stk_font_gen: create a custom	stk_font_gen(1)
stroke font	storing strings /unstr: create	strfile(6)
a random access file for	stream editor	sed(1)
sed:	strfile, unstr: create a	strfile(6)
random access file for/		

strings in an object file	strings: find the printable	strings(1)
implement/ xstr: extract	strings from C programs to	xstr(1)
strings: find the printable	strings in an object file	strings(1)
random access file for storing	strings /unstr: create a	strfile(6)
C programs to implement shared	strings /extract strings from	xstr(1)
basename:	strip filename affixes	basename(1)
relocation bits	strip: remove symbols and	strip(1)
stk_font_gen: create a custom	stroke font	stk_font_gen(1)
programs	struct: structure FORTRAN	struct(1)
struct:	structure FORTRAN programs	struct(1)
characteristics of a document	stty: set terminal options	stty(1)
temporarily	style: analyze surface	style(1)
su:	su: substitute user ID	su(1)
subsystem	substitute user ID temporarily	su(1)
sccs: front end for the SCCS	subsystem	sccs(1)
sum:	sum and count blocks in a file	sum(1)
file	sum: sum and count blocks in a	sum(1)
du:	summarize disk usage	du(1)
document style: analyze	surface characteristics of a	style(1)
interval sleep:	suspend execution for an	sleep(1)
ownership of the/ xownroot:	swapul: rearrange underlining	swapul(1)
display address of external	switch between X and DM	xownroot(1)
strip: remove	symbol esa:	esa(1)
interpreter) with C-like	symbols and relocation bits	strip(1)
tbl: format	syntax csh: a shell (command	csh(1)
tabs: set terminal	systype: display version stamp	systype(1)
expand, unexpand: expand	tables for nroff or troff	tbl(1)
ctags: create a	tabs: set terminal tabs	tabs(1)
a file	tabs	tabs(1)
talk:	tabs to spaces, and vice versa	expand(1)
tar:	tags file	ctags(1)
mt: magnetic	tail: deliver the last part of	tail(1)
tcopy: copy a mag	talk: talk to another user	talk(1)
deroff: remove nroff, troff,	talk to another user	talk(1)
or troff	tar: tape archiver	tar(1)
teachgammon:	tape manipulating program	mt(1)
backgammon	tape	tcopy(1)
last logins of users and	tar: tape archiver	tar(1)
telnet: user interface to the	tb: print process traceback	tb(1)
TELNET protocol	tbl, and eqn constructs	deroff(1)
su: substitute user ID	tbl: format tables for nroff	tbl(1)
terminals	tcopy: copy a mag tape	tcopy(1)
flake: induce	teach the game of backgammon	teachgammon(6)
initialization tset:	teachgammon: teach the game of	teachgammon(6)
emt: emulate a dumb	tee: pipe fitting	tee(1)
xterm:	teletypes last: indicate	last(1)
vt100: VT100	TELNET protocol	telnet(1C)
tty: get	telnet: user interface to the	telnet(1C)
	temporarily	su(1)
	term: conventional names for	term(7)
	terminal dandruff	flake(6)
	terminal dependent	tset(1)
	terminal	emt(1)
	terminal emulator for X	xterm(1)
	terminal emulator	vt100
	terminal name	tty(1)

stty: set	terminal options	stty(1)
clear: clear	terminal screen	clear(1)
script: make typescript of	terminal session	script(1)
tabs: set	terminal tabs	tabs(1)
animate worms on a display	terminal worms:	worms(6)
term: conventional names for	terminals	term(7)
kill:	terminate or signal a process	kill(1)
trty:	test a type trait	trty(1)
	test: condition command	test(1)
quiz:	test your knowledge	quiz(6)
ed:	text editor	ed(1)
ex, edit:	text editor	ex(1)
checkeq: format mathematical	text eqn, neqn,	eqn(1)
fmt: simple	text formatter	fmt(1)
typesetting troff:	text formatting and	troff(1)
ms:	text formatting macros	ms(7)
nroff:	text formatting	nroff(1)
translate status code value to	text message stcode:	stcode(1)
protocol	tftp: trivial file transfer	tftp(1C)
/print wordy sentences;	thesaurus for diction	diction(1)
comparison diff3:	three-way differential file	diff3(1)
ttt:	tic-tac-toe	ttt(6)
time:	time a command	time(1)
built bldt: display	time operating system was	bldt(1)
display the module name and	time stamp ts:	ts(1)
	time: time a command	time(1)
tz: set or display system	time zone	tz(1)
scrtio: set/show screen	timeout	scrtio(1)
system	tip, cu: connect to a remote	tip(1C)
tsort:	topological sort	tsort(1)
modified of a file	touch: update date last	touch(1)
tpm: set/display	touchpad and mouse/	tpm(1)
mouse characteristics	tpm: set/display touchpad and	tpm(1)
	tr: translate characters	tr(1)
tb: print process	traceback	tb(1)
trty: test a type	trait	trty(1)
crpad: create a	transcript pad and window	crpad(1)
execute a/ dmwin: create a DM	transcript/input pad pair and	dmwin(1)
ftp: ARPANET file	transfer program	ftp(1C)
tftp: trivial file	transfer protocol	tftp(1C)
tr:	translate characters	tr(1)
text message stcode:	translate status code value to	stcode(1)
within a font tr_font:	transliterate characters	tr_font(1)
/a binary file for	transmission via mail	uencode(1C)
X - a portable, network	transparent window system	X(1)
	trek: trekkie game	trek(6)
trek:	trekkie game	trek(6)
characters within a font	tr_font: transliterate	tr_font(1)
tftp:	trivial file transfer protocol	tftp(1C)
deroff: remove nroff,	troff, tbl, and eqn constructs	deroff(1)
format tables for nroff or	troff tbl:	tbl(1)
typesetting	troff: text formatting and	troff(1)
battlestar: a	tropical adventure game	battlestar(6)
	trty: test a type trait	trty(1)
values	true, false: provide truth	true(1)
false,	true: provide truth values	false(1)

prism: get processor type	truth value for prism	prism(1)
false, true: provide	truth values	false(1)
true, false: provide	truth values	true(1)
and time stamp	ts: display the module name	ts(1)
initialization	tset: terminal dependent	tset(1)
	tsort: topological sort	tsort(1)
	tt: tic-tac-toe	ttt(6)
	tty: get terminal name	tty(1)
crtty: create a new	type	crtty(1)
dlty: delete a	type	dlty(1)
file: determine file	type	file(1)
intty: install a new	type	intty(1)
intm: install a	type manager	intm(1)
crttyobj: create a	type object module for binding	crttyobj(1)
trty: test a	type trait	trty(1)
prism: get processor	type truth value for prism	prism(1)
lty: list installed	types	lty(1)
script: make	typescript of terminal session	script(1)
man: macros to	typeset manual	man(7)
troff: text formatting and	typesetting	troff(1)
zone	tz: set or display system time	tz(1)
	ul: do underlining	ul(1)
uncompress files/ compact,	uncompact, ccat: compress and	compact(1)
/uncompact, ccat: compress and	uncompress files, and then cat/	compact(1)
expand data compress,	uncompress, zcat: compress and	compress(1)
swapul: rearrange	underlining	swapul(1)
ul: do	underlining	ul(1)
file unget:	undo a previous get of an SCCS	unget(1)
spaces, and vice/ expand,	unexpand: expand tabs to	expand(1)
an SCCS file	unget: undo a previous get of	unget(1)
	unifdef: remove ifdef'ed lines	unifdef(1)
a file	uniq: report repeated lines in	uniq(1)
	units: conversion program	units(1)
execution uux:	UNIX-to-UNIX system command	uux(1)
uucp, uulog, uuname:	UNIX-to-UNIX system copy	uucp(1C)
uuto, uupick: public	UNIX-to-UNIX system file copy	uuto(1)
rmdir, rm: remove	(unlink) directories or files	rmdir(1)
rm, rmdir: remove	(unlink) files or directories	rm(1)
file for storing/ strfile,	unstr: create a random access	strfile(6)
or groups of programs make:	update and regenerate programs	make(1)
file touch:	update date last modified of a	touch(1)
/convert pathnames between	upper and lowercase and/	cvtnam(1)
has been up	uptime: show how long system	uptime(1)
du: summarize disk	usage	du(1)
miscellaneous: miscellaneous	useful information pages	intro(7)
environ:	user environment variables	environ(7)
su: substitute	user ID temporarily	su(1)
print effective current	user id whoami:	whoami(1)
program finger:	user information lookup	finger(1)
protocol telnet:	user interface to the TELNET	telnet(1C)
xset -	user preference utility for X	xset(1)
talk: talk to another	user	talk(1)
write: write to another	user	write(1)
whois: DARPA Internet	username directory service	whois(1)
last: indicate last logins of	users and teletypes	last(1)
send or receive mail among	users binmail:	binmail(1)

who are on a node	users: compact list of users	users(1)
wall: write to all	users	wall(1)
users: compact list of	users who are on a node	users(1)
inlib: install a	user-supplied library	inlib(1)
xset - user preference	utility for X	xset(1)
remote mail received via	uucp rmail: handle	rmail(1)
control uustat:	uucp status inquiry and job	uustat(1)
UNIX-to-UNIX system copy	uucp, uulog, uuname:	uucp(1C)
encode/decode a binary file/	uuencode, uudecode:	uuencode(1C)
system copy uucp,	uulog, uuname: UNIX-to-UNIX	uucp(1C)
copy uucp, uulog,	uuname: UNIX-to-UNIX system	uucp(1C)
system file copy uuto,	uupick: public UNIX-to-UNIX	uuto(1)
and job control	uustat: uucp status inquiry	uustat(1)
UNIX-to-UNIX system file copy	uuto, uupick: public	uuto(1)
command execution	uux: UNIX-to-UNIX system	uux(1)
Manager Commands .if '. dm(1)	uwm - a window manager for X	uwm(1)
	v'2' dm/commands - Display	dm(1)
	val: validate SCCS file	val(1)
	val: validate SCCS file	val(1)
get processor type truth	value for prism prism:	prism(1)
stcode: translate status code	value to text message	stcode(1)
false, true: provide truth	values	false(1)
true, false: provide truth	values	true(1)
environ: user environment	variables	environ(7)
	vc: version control	vc(1)
shell commands	ver: change the version of	ver(1)
lint: a C program	verifier	lint(1)
tabs to spaces, and vice	versa /unexpand: expand	expand(1)
vc:	version control	vc(1)
get: get a	version of an SCCS file	get(1)
ver: change the	version of shell commands	ver(1)
hangman: Computer	version of the game hangman	hangman(6)
systype: display	version stamp	systype(1)
scsdiff: compare two	versions of an SCCS file	scsdiff(1)
were used to/ what: show what	versions of object modules	what(1)
programs	vgrind: grind nice listings of	vgrind(1)
display editor based on ex	vi: screen oriented (visual)	vi(1)
a binary file for transmission	via mail /encode/decode	uuencode(1C)
handle remote mail received	via uucp rmail:	rmail(1)
expand tabs to spaces, and	vice versa expand, unexpand:	expand(1)
file perusal filter for CRT	viewing more, page:	more(1)
robots: fight off	villainous robots	robots(6)
	vine: grow vines	vine(6)
vine: grow	vines	vine(6)
on ex vi: screen oriented	(visual) display editor based	vi(1)
window settings	vsize: set/display VT100	vsize(1)
vt100:	VT100 terminal emulator	vt100
	vt100: VT100 terminal emulator	vt100
vsize: set/display	VT100 window settings	vsize(1)
process	wait: await completion of	wait(1)
atq: print the queue of jobs	waiting to be run	atq(1)
	wall: write to all users	wall(1)
backup file	wbak: create a magnetic media	wbak(1)
	wc: word count	wc(1)
whatis: describe	what a command is	whatis(1)
object modules were used to/	what: show what versions of	what(1)

modules were used/ what: show	what versions of object	what(1)
command is	whatis: describe what a	whatis(1)
binary, and/or manual for/	whereis: locate source,	whereis(1)
users: compact list of users	who are on a node	users(1)
from:	who is my mail from?	from(1)
who:	who is on the system	who(1)
notified if mail arrives and	who it is from biff: be	biff(1)
current user id	who: who is on the system	who(1)
directory service	whoami: print effective	whoami(1)
machines rwho:	whois: DARPA Internet username	whois(1)
create a transcript pad and	who's logged in on local	rwho(1C)
uwm - a	window crpad:	crpad(1)
vsize: set/display VT100	window manager for X	uwm(1)
Apollo Domain/X11 shared X	window settings	vsize(1)
X - X	window system server Xapollo:	Xapollo(1)
portable, network transparent	Window System server	Xserver(1)
sail: multi-user	window system X - a	X(1)
diction, explain: print	wooden ships and iron men	sail(6)
cd: change	wordy sentences; thesaurus for/	diction(1)
pwd:	working directory	cd(1)
worm: play the growing	working directory name	pwd(1)
game	worm game	worm(6)
display terminal	worm: play the growing worm	worm(6)
worms: animate	worms: animate worms on a	worms(6)
wall:	worms on a display terminal	worms(6)
write:	write to all users	wall(1)
hunt-the-wumpus	write to another user	write(1)
transparent window system	write: write to another user	write(1)
xownroot: switch between	wump: the game of	wump(6)
of the root while the	X - a portable, network	X(1)
uwm - a window manager for	X - X Window System server	Xserver(1)
/Apollo Domain/X11 shared	X and DM ownership of the root/	xownroot(1)
X -	X server is running /ownership	xownroot(1)
- user preference utility for	X	uwm(1)
xterm: terminal emulator for	X window system server	Xapollo(1)
- BDF to SNF font compiler for	X Window System server	Xserver(1)
shared X window system/	X xset	xset(1)
from the shell	X	xterm(1)
DM ownership of the root/	X11 bdf2snf	bdf2snf(1)
for X	Xapollo: Apollo Domain/X11	Xapollo(1)
programs to implement shared/	xdmcc: execute a DM command	xdmcc(1)
compiler-compiler	xownroot: switch between X and	xownroot(1)
affirmative	xset - user preference utility	xset(1)
compress, uncompress,	xstr: extract strings from C	xstr(1)
tz: set or display system time	xterm: terminal emulator for X	xterm(1)
	yacc: yet another	yacc(1)
	yes: be repetitively	yes(1)
	zcat: compress and expand data	compress(1)
	zone	tz(1)

the 1990s, the number of people in the UK who are employed in the public sector has increased by 1.5 million, from 2.5 million in 1980 to 4 million in 1995. The public sector has become a major employer in the UK, and its growth has been a key factor in the overall growth of the economy.

The public sector has also become a major employer in the UK, and its growth has been a key factor in the overall growth of the economy. The public sector has become a major employer in the UK, and its growth has been a key factor in the overall growth of the economy.

The public sector has become a major employer in the UK, and its growth has been a key factor in the overall growth of the economy. The public sector has become a major employer in the UK, and its growth has been a key factor in the overall growth of the economy.

The public sector has become a major employer in the UK, and its growth has been a key factor in the overall growth of the economy. The public sector has become a major employer in the UK, and its growth has been a key factor in the overall growth of the economy.

The public sector has become a major employer in the UK, and its growth has been a key factor in the overall growth of the economy. The public sector has become a major employer in the UK, and its growth has been a key factor in the overall growth of the economy.

The public sector has become a major employer in the UK, and its growth has been a key factor in the overall growth of the economy. The public sector has become a major employer in the UK, and its growth has been a key factor in the overall growth of the economy.

The public sector has become a major employer in the UK, and its growth has been a key factor in the overall growth of the economy. The public sector has become a major employer in the UK, and its growth has been a key factor in the overall growth of the economy.

The public sector has become a major employer in the UK, and its growth has been a key factor in the overall growth of the economy. The public sector has become a major employer in the UK, and its growth has been a key factor in the overall growth of the economy.

The public sector has become a major employer in the UK, and its growth has been a key factor in the overall growth of the economy. The public sector has become a major employer in the UK, and its growth has been a key factor in the overall growth of the economy.

Section 1

Commands

NAME

intro – introduction to commands

DESCRIPTION

This section describes publicly accessible commands in alphabetic order. Certain distinctions of purpose are made in the headings, as shown below:

- (1) Commands of general utility.
- (1C) Commands for communication with other systems.
- (1G) Commands used primarily for graphics and computer-aided design.

Commands related to system maintenance used to appear in section 1 manual pages and were distinguished by (1M) at the top of the page. These manual pages now appear in section 8.

DIAGNOSTICS

Upon termination, each command returns two bytes of status. One byte is supplied by the system giving the cause for termination, and (in the case of "normal" termination) one is supplied by the program; see **wait(1)**, and **exit(2)**. The former byte's value is 0 for normal termination, the latter is customarily 0 for successful execution, nonzero to indicate troubles such as erroneous parameters, bad or inaccessible data, or other inability to cope with the task at hand. It is called variously "exit code", "exit status", or "return code", and is described only where special conventions are involved.

NOTE

Some commands can exhibit undefined behavior when processing 8-bit character data. Caution is suggested, as in some cases the 8-bit data will not be preserved.

SEE ALSO

domain(1)
Section 6 (for computer games),
Getting Started with Domain/OS, and
Using Your BSD Environment.

NAME

domain – Domain/OS-specific commands and extensions

DESCRIPTION

While providing all of the significant functionality of 4.3BSD, Domain/OS BSD actually represents only a subset of the greater functionality of Domain/OS. Furthermore, Domain/OS BSD omits some features of 4.3BSD that are irrelevant to Apollo workstations. The following paragraphs describe aspects of that greater functionality that are visible to the Domain/OS BSD user and summarize features of 4.3BSD not implemented under Domain/OS BSD.

Domain/OS Additions to the BSD Environment

Pages that describe the Domain/OS-specific commands are identified with the page heading “Domain/OS BSD;” pages describing standard 4.3BSD commands simply say “BSD”.

The /usr/apollo/bin Directory

All systems, even if they only have the Aegis environment, now have a **/usr/apollo** directory. It contains certain Domain/OS extensions to the UNIX environment. It also includes C include files for Domain/OS system calls, as well as other added-value files.

The **/usr/apollo/bin** directory contains Domain/OS commands that apply to all three environments or extend the UNIX environment. The following Domain/OS-specific commands appear in **/usr/apollo/bin**

bldt	Displays information about the version of Domain/OS.
chacl	Changes the entries in an object's access control list (ACL).
cpacl	Copies access control lists (ACLs).
cpscr	Copies the current screen image, without clearing it, to the file you specify.
crddf	Creates, displays, or modifies a device descriptor file (DDF).
crp	Creates a process on a remote node.
crpad	Creates a transcript pad, copies a file (or standard input) into that pad, and then opens a window into the pad.
crtv	Creates a new type. It creates an identifier for the new type, and associates it with the supplied type name. New types are used to identify a new kind of manager for streams.
crtvobj	Creates an object module that contains a global symbol with the type UID. This module is bound with type managers. The variable is passed into calls to trait_\$mgr_dcl to declare support for the specified type.
cvt_font	Converts SR9 font files to the new font file format for SR10.

cvtname	Converts pathnames between uppercase and lowercase and preserves colons.
cvtrgy	Allows the system administrator to generate an SR10 format registry database from SR9.7 registry files, or generates SR9.7 registry files with data from the SR10 registry.
dbacl	Provides an interactive menu-based editor for manipulating Access Control Lists (ACLs).
dde	Allows you to load and debug programs written in any programming language supported by the Domain/OS operating system, including assembler.
dlty	Deletes a type and any installed type manager.
dm	Contains a list of Display Manager commands.
dspst	Displays process statistics in a graphical, bar-chart fashion within the current process window.
edfont	An interactive program that allows you to create, edit and view character font files.
edmtdesc	Allows you to create, list, and modify the magnetic tape descriptor object.
emt	Allows your node to emulate an ASCII terminal connected to another computer. This asynchronous connection exists through a stream opened on one of the node's SIO lines. emt also permits ASCII file transfer between your node and the remote host.
esa	Displays the address of an external symbol in an installed library. This command is primarily intended for system-level debugging.
fst	Prints information about the most recent fault that occurred in the process.
hpc	The hpc (histogram_program_counter) command, part of Domain/PAK (Domain Performance Analysis Kit), looks at the performance of programs at the program counter (PC) level.
intm	Installs a type manager for the <i>type_name</i> .
inty	Installs a type from one node to another.
iso	Converts files written with the overloaded 7-bit national fonts to the International Standards Organization (ISO) 8-bit format. The iso commands include: french_to_iso , german_to_iso , nor.dan_to_iso , swedish_to_iso , swiss_to_iso , and uk_to_iso .

kbm	Allows you to set and display the characteristics for the keyboard.
las	Produces a list of objects mapped into the address space.
lbr2ar	Converts pre-SR10 lbr library files containing object modules in OBJ format to SR10 ar library archive files containing object modules in COFF format.
lcm	Loads a color map from a file that specifies a set of color map entries.
llib	Lists those libraries which have been installed in the current process via the build-in inlib shell.
llkob	Lists the locked objects resident on volumes mounted on this node, and objects resident in other nodes that are locked by processes running locally.
lsacl	Shows the access control list (ACL) associated with the files and directories specified.
lty	Lists the types currently installed on a volume.
mkapr	Makes an Apollo product report.
obj2coff	Converts SR9.5 or later object format modules to SR10 COFF format modules. Either individual modules, or complete bound programs may be converted.
prf	Queues a file for printing.
rbak	Restores objects from the backup input media written by wbak (write_backup). The backup input media can be magnetic media, file or standard input.
rwmt	Reads tapes from non-Domain installations and writes tapes that can be read by non-Domain installations. rwmt can read and write unlabeled tapes, as well as ANSI level 1-4 labeled tapes.
scrattr	Lists the <i>X</i> and <i>Y</i> dimensions of the display in pixels.
scrt0	Sets or displays the number of minutes the system waits before it shuts off the display screen. It begins counting minutes after the last input event or window configuration change.
stcode	Prints the text message associated with a hexadecimal status code.
systype	Displays the UNIX version stamp of the specified object file.
tb	Prints a process traceback, listing the name and current line number of each routine on the call stack.
tpm	Allows you to define and display characteristics for the touchpad and mouse.

tr_font	Allows you to change the order in which characters appear in fonts.
trty	Tests whether some type supports a given trait. It prints a message and returns a status that a shell can test
ts	Displays the time stamp and module name stored in an object module.
tz	Sets the system time zone to a known time zone or to an offset from Coordinate Universal Time (utc).
vsize	Allows you to set the dimensions of the VT100* emulator window pane. This command is valid only from within the VT100 emulator (which is invoked with the VT100 command); attempting to use it directly from the shell causes an error.
vt100	Creates a window running the VT100 terminal emulator and starts up a shell within the window.
wbak	Writes one or more objects to either a removable media, disk file or standard output.
xdmc	Allows you to invoke Display Manager commands from the command shell or from within a shell script.

Domain/OS BSD Extensions

This section describes Domain/OS extensions to standard UNIX commands.

ar	Domain/OS BSD ar builds a module name table and a long name table in addition to the symbol table; these tables are stored in files that are never mentioned or accessible. This makes ranlib obsolete.
cc	cc is the Domain/OS interface to the preprocessor (cpp), the Domain C compiler, and the link editor (ld). The Domain/OS cc command provides some unique options; not all standard UNIX options are available. The -A option identifies a unique set of Domain/OS extensions to cc and ld .
coffdump	coffdump is the Domain/OS implementation of an extended version of the System V Release 3 dump command. Domain/OS does not support the 4.3BSD dump command.
cp	The cp command includes a number of Domain/OS extensions. See cp(1) for a complete description of these extensions.
dbx	dbx creates a separate transcript pad for debugger interactions unless you specify the -no_frame option. dbx also creates a window to display source code unless you specify the -no_src option.
df	The nodename argument allows you to print the amount of free disk space available on the specified node, nodename .

f77	f77 is the Domain/OS interface to the preprocessors, the Domain FTN compiler, and the link editor (ld (1)). The Domain/OS BSD f77 command provides some unique options; not all standard 4.3BSD options are available.
file	The BSD environment uses the System V Release 3 version of the file command.
ksh	Domain/OS BSD supports the Korn shell and adds some extensions to this shell. See "Extensions to the UNIX Shells," below, for a brief summary of the added features; see ksh (1) for complete information.
ld	The Domain/OS version of ld includes support for features that are not available on 4.3BSD. Domain/OS ld supports the following extensions: static resource information records (.sri), module information records (.mir), and control of global variable visibility. The -A option identifies a unique set of Domain/OS BSD extensions to cc and ld .
login	The /bin/login command is a merge of the System V and 4.3BSD login commands. The environment can be expanded or modified by supplying additional arguments to login, either at execution time or when login requests your log-in name. Variables containing = are placed in the environment without modification. Domain/OS login includes new security features for dial-up lines, /etc/d_users and /etc/d_passwd. /etc/d_users is simply a file containing a list of users authorized to log in on this node. /etc/d_passwd is a file containing lines which specify a user's log-in shell, and the dial-in password for the specified shell as returned by crypt(3). See login(1) for more information.
ln	Symbolic links in Domain/OS are implemented as soft links. These are identical in behavior, except that soft links do not have protections associated with the links themselves.
lorder	The need for lorder has vanished on Domain/OS systems, since ar (1) and ld (1) cooperate to create randomly accessed libraries.
ls	If you specify -T with the -l option, ls shows the Domain/OS "type" of each file.
mkdir	The mechanism for assigning the initial file ACL and initial directory ACL for the directory created by the mkdir command has been changed. The initial file ACL and initial directory ACL are now inherited from the parent directory.
nm	The -Ag option checks KGT (Known Global Table) to see if undefined globals are defined in global libraries. If specified with the -u option, nm will not print those undefined symbols that are defined in global libraries.

passwd	On Domain/OS systems, the <code>/etc/passwd</code> file is a typed file, which is automatically generated by the registry daemon. The registry administrator can make the person information in the registry read-only, in which case normal users cannot change the "Name" field.
ps	The Domain/OS <i>nodename</i> option shows information about processes running on the specified node.
ranlib	ranlib is not necessary on Domain/OS systems; however, it is provided as a no-op for compatibility.
strip	The <code>-Aa</code> option strips all debugging information, including that needed for traceback. The <code>.blocks</code> and <code>.lines</code> sections, if present, will be removed. This option strips more information than the default strip behavior, and is added for users who wish to remove all symbolic information.
tftp	The Domain/OS versions of tftp (1C) and tftpd (8C) are adaptations of the MIT Project Athena implementations of the tftp protocol. Domain/OS tftp will interface with any RFC783 compliant implementation.
uucp	Domain/OS supports "HoneyDanBer" UUCP for both the BSD and SysV environments. (See "UUCP Support" below).
who	The who command includes a number of Domain/OS extensions. See who (1) for a complete description of these extensions.

Extensions to the UNIX Shells

Domain/OS includes support in **cs****h**, **ksh**, and **sh** for the built-in commands **inlib**, **rootnode**, and **ver**.

rootnode	Causes <code>/</code> to refer to the node entry directory of <i>nodename</i> instead of the current node entry directory.
inlib	Installs a library at the current shell level; it remains installed until the shell that installed it exits. The newly installed library will be used to resolve external references of programs (and libraries) loaded after its installation.
ver	Changes, temporarily or permanently, the UNIX version of commands that are executed by the shell. This command also displays the version in use.

cs**h** and **sh** also include support for a new command line option, `-Dname=value`. This option sets the parameter *name* to *value*, then passes it to the shell's environment. This option is useful for tailoring the environment of a shell invoked from a program that isn't a shell (such as the Display Manager).

ksh has been extended to support editing commands in Display Manager pads. If the value of the variable **EDITOR** ends in **emacs**, **gmacs**, or **vi** and the **VISUAL** is not set, the corresponding option is turned on. This value should be unset for shells running in Apollo transcript pads.

For Apollo transcript pads, the variable **FCEDIT** should be set to **pad**. With dial-up lines or in VT100 windows, values like **vi** or **emacs** are useful.

The in-line editing options are not useful in Apollo transcript pads. The command input pane associated with transcript pads allows full command line editing. Setting **VISUAL** or **EDITOR** in Apollo transcript pads causes the pad to flip in and out of raw mode.

UUCP Support

At SR10, Domain/OS supports only the System V HoneyDanBer version of UUCP in both the BSD and SysV environments. The following BSD commands have been replaced by their System V HoneyDanBer equivalents:

- uucico**(8C)
- uuclean**(8C)
- uucp**(1C)
- uucpd**(8C)
- uulog**(1C)
- uuname**(1C)
- uupoll**(8C)
- uuq**(1C)
- uusend**(1C)
- uusnap**(8C)
- uux**(1C)
- uuxqt**(8C)

The following HoneyDanBer UUCP commands have been added to the BSD environment

- uucheck**(8C)
- uucleanup**(8C)
- uusched**(8C)
- uustat**(1C)
- uuto**(1C)
- uux**(1C)

Unsupported Commands

The following commands from 4.3BSD are not supported.

adb	mset	systat
as	pc	tc
crypt	pdx	tk
efl	pi	tn3270
fp	pix	tp
gcore	pmerge	vmstat
iostat	px	vwidth
learn	pxp	w
llsp	pxref	window
liszt	sendbug	xsend
lock	symorder	xget, enroll
lxref	sysline	

Manual pages describing specific features may point out other differences and unsupported features. These descriptions usually appear in the "NOTES" section.

SEE ALSO

intro(1), domain(2), domain(3), domain(7), domain(8),
Using Your BSD Environment;
Managing BSD System Software.

NAME

addbib – create or extend bibliographic database

SYNOPSIS

addbib [**-p** *promptfile*] [**-a**] *database*

DESCRIPTION

When the **addbib** program starts up, answering “y” to the initial “Instructions?” prompt yields directions; typing “n” or <RETURN> skips them. **addbib** then prompts for various bibliographic fields, reads responses from the terminal, and sends output records to a *database*. A null response (just <RETURN>) means leave out that field. A minus sign (–) means go back to the previous field. A trailing backslash allows a field to be continued on the next line. The repeating “Continue?” prompt allows you to do the following:

- Resume by typing “y” or <RETURN>
- Quit the current session by typing “n” or “q”
- Edit the *database* with any system editor (**vi**(1), **ex**(1), **edit**(1), **ed**(1))

OPTIONS

-a Suppress prompting for an abstract; asking for an abstract is the default. Abstracts are ended with a CTRL/D.

-p *promptfile* Use a new prompting skeleton, defined in *promptfile*. This file should contain prompt strings, a tab, and the keyletters to be written to the *database*.

BIBLIOGRAPHIC FIELDS

The bibliographic fields used by **addbib** consist of various keyletters. The most common keyletters and their meanings are listed below. **addbib** insulates you from these keyletters, since it prompts you in English; but if you edit the bibliography file later on, you need to know this information.

%A	Author's name
%B	Book containing article referenced
%C	City (place of publication)
%D	Date of publication
%E	Editor of book containing article referenced
%F	Footnote number or label (supplied by refer (1))
%G	Government order number
%H	Header commentary, printed before reference
%I	Issuer (publisher)
%J	Journal containing article
%K	Keywords to use in locating reference
%L	Label field used by -k option of refer
%M	AT&T Bell Labs Memorandum (undefined)
%N	Number within volume
%O	Other commentary, printed at end of reference
%P	Page number(s)
%Q	Corporate or Foreign Author (unreversed)

- %R Report, paper, or thesis (unpublished)
- %S Series title
- %T Title of article or book
- %V Volume number
- %X Abstract — used by **roffbib**, not by **refer**
- %Y,Z Ignored by **refer**

Except for “%A”, you should provide each relevant field only once. For example,

- %A Bill Tuthill
- %T Refer — A Bibliography System
- %I Computing Services
- %C Berkeley
- %D 1982
- %O UNX 4.3.5.

FILES

promptfile Optional file to define prompting

SEE ALSO

refer(1), **sortbib(1)**, **roffbib(1)**, **indxbib(1)**, **lookbib(1)**

NAME

admin – create and administer SCCS files

SYNOPSIS

admin [*options*] *file* ...

DESCRIPTION

The **admin** command creates new source code control system (SCCS) files and changes the parameters of existing ones. You can specify several different options (in any order) along with named files. An *s* followed by a period, must prefix all SCCS filenames (for example, *s.file1*).

If a named *file* does not exist, **admin** creates it. It then initializes the file's parameters according to specified options. Parameters not initialized by a valid option, discussed below, are assigned a default value. If a named *file* does exist, **admin** changes the file's parameters that correspond to specified options; other parameters are left as is.

If you have named a directory in the command line, **admin** behaves as though each file in the directory were a named file, except that it silently ignores non-SCCS and unreadable files. If you give a dash (–) in place of the filename, **admin** reads the standard input, taking each line to be the name of an SCCS file to be processed. Again, it silently ignores non-SCCS and unreadable files.

OPTIONS

Each option below assumes that only one named file is to be processed, since the effects of the arguments apply independently to each named file. On options beginning with **–f** and followed by an argument, the specified flag (and value, if applicable) is placed in the SCCS file. You can supply several of these **–f** options on a single **admin** command line.

- n** Create a new SCCS file from the contents of the file specified.
- i[*name*]** Specify *name* as the name of the file from which the text for a new SCCS file is to be taken. The text comprises the first delta of the specified file. If you omit *name*, **admin** gets the text for the SCCS file by reading the standard input until it encounters an end-of-file. This option allows you to create only one SCCS file at a time. Without it, the SCCS file is created empty.
- rrel** Insert the initial delta into the specified release (the default is release 1). You can use this option only if the **–i** option is also present. Note that the default level of initial deltas is always 1; thus, the naming of initial deltas defaults to 1.1.
- t[*name*]** Use the name of the file specified for descriptive text for the SCCS file. If you use this option with **–n** and/or **–i**, you must also supply the descriptive text filename. For existing SCCS files, using this option without specifying a filename removes any descriptive text from the SCCS file. If you specify a filename, any descriptive text in the SCCS file is replaced by existing text in the named file.

- fb** Allow use of the **-b** keyletter on a **get(1)** command to create branch deltas.
- fcceil** Specify the highest release ("ceiling") that may be retrieved by a **get(1)** command for editing. The number should be less than or equal to 9999. The default is 9999.
- fffloor** Specify the lowest release ("floor") that may be retrieved by a **get(1)** command for editing. The number should be greater than zero but less than 9999. The default is 1.
- fdSID** Indicate the default delta number (SID) to be used by a **get(1)** command.
- fi[*str*]** Treat the "No id keywords (ge6)" message issued by **get(1)** or **delta(1)** as a fatal error. In the absence of this flag, the message is only a warning. The message is issued if no SCCS identification keywords are found in the text retrieved or stored in the SCCS file. If you supply a value, the keywords must be identical to the given string; however, the string must contain a keyword, and must not contain embedded newlines.
- fj** Allow concurrent **get(1)** commands for editing on the same (SID) of an SCCS file. This allows multiple concurrent updates to the same version of the SCCS file.
- flist** Specify a *list* of releases to which deltas can no longer be made (**get -e** against one of these "locked" releases fails). The *list* has the following syntax:


```
<list> ::= <range> | <list> , <range>
<range> ::=      RELEASE NUMBER | a
```

The character **a** in the *list* is equivalent to specifying all releases for the named SCCS file.
- fn** Force **delta(1)** to create a "null" delta in any release being skipped when a delta is made in a new release (for example, in making delta 5.1 after delta 2.7, releases 3 and 4 are skipped). These null deltas serve as anchor points so that branch deltas may later be created from them. The absence of this flag causes skipped releases to be non-existent in the SCCS file, preventing branch deltas from being created from them in the future.
- fqtext** Substitute user-definable text for all occurrences of the **%Q%** keyword in SCCS file text retrieved by **get(1)**.
- fmmod** Substitute the module name of the SCCS file for all occurrences of the **%M%** keyword in SCCS file text retrieved by **get(1)**. If you do not specify the **-m** option, the value assigned is the name of the SCCS file with the leading **s.** removed.

- ftype** Substitute the *type* of module in the SCCS file for all occurrences of %Y% keyword in SCCS file text retrieved by **get(1)**.
- fv[pgm]** Force **delta(1)** to prompt for modification request (MR) numbers as the reason for creating a delta. The optional value specifies the name of an MR number validity-checking program. (If this flag is set when creating an SCCS file, you must also use the **-m** option, even if its value is null.)
- dflag** Remove (delete) the specified *flag* from an SCCS file. You can specify the **-d** option only when processing existing SCCS files. You can supply several **-d** options on a single **admin** command. See the options beginning with a **-f** (for example, **-fb**, **-fj**) for allowable *flag* names.
- l~~ist~~** Specify a *list* of releases to be "unlocked." See the **-fl~~ist~~** option for a description of the **l** flag and the syntax of a *list*.
- a~~login~~** Indicate a login name, or numerical UNIX system group ID, to be added to the list of users that can make deltas (changes) to the SCCS file. A group ID is equivalent to specifying all login names common to that group ID. You can specify several of these options on a single **admin** command line. You can put as many logins, or numerical group IDs, as you want on the list simultaneously. If the list of users is empty, anyone can make changes. A user with a login or group ID preceded by an exclamation point (!) is not allowed to make changes.
- e~~login~~** Indicate a login name, or numerical group ID, to be erased from the list of users allowed to make deltas (changes) to the SCCS file. Specifying a group ID is equivalent to specifying all login names common to that group ID. You can specify several **-e** options on a single **admin** command line.
- y[comment]** Insert the *comment* text into the SCCS file as a comment for the initial delta in a manner identical to that of **delta(1)**. Omitting the **-y** option results in a default comment line being inserted in the form
- date and time created YY/MM/DD HH:MM:SS by *login*
- The **-y** option is valid only if you specify the **-i** and/or **-n** options (that is, if you are creating a new SCCS file).
- m[mrlist]** Insert the list of MR numbers into the SCCS file as the reason for creating the initial delta in a manner identical to **delta(1)**. The **v** flag must be set and the MR numbers are validated if the **v** flag has a value (the name of an MR number validation program). Diagnostics occur if the **v** flag is not set or MR validation fails. See the **-fv** option described above.

- h** Check the structure of the SCCS file and compare a newly computed check-sum (the sum of all the characters in the SCCS file except those in the first line) with the check-sum that is stored in the first line of the SCCS file. This option produces appropriate error diagnostics. It also inhibits writing on the file, so that it nullifies the effect of any other options supplied. Therefore, it is meaningful only when processing existing files.
- z** Recompute the SCCS file check-sum and store it in the first line of the SCCS file (refer to the **-h** option above). Note that using this option on a truly corrupted file may prevent future detection of the corruption.

EXAMPLE

To create a new SCCS file (called **sccsfile1**) from **file1**, type the following:

```
# admin -ifile1 s.sccsfile1
```

BUGS

Directories containing SCCS files should be 755 mode, so that only the owner can modify SCCS contained in the directories. The mode of SCCS files themselves should be 444, to prevent modifications except by SCCS commands.

If you have to patch an SCCS file, the owner must change the file's mode to 644, to allow use of an editor. Always process the edited file by an **admin -h** to check for corruption, followed by an **admin -z** to generate a proper check-sum. You should then specify another **admin -h** to ensure that the SCCS file is valid.

DIAGNOSTICS

Use **help(1)** for explanations.

SEE ALSO

delta(1), **get(1)**, **help(1)**, **prs(1)**, **sccs(1)**, **what(1)**, **sccsfile(4)**

NAME

apply – apply a command to a set of arguments

SYNOPSIS

apply [**-ac**] [**-n**] *command args* ...

DESCRIPTION

apply runs the named *command* on each argument in turn. Normally you choose arguments singly.

OPTIONS

- n** Specifies the number of arguments to be passed to *command*. If *n* is zero, *command* is run without arguments once for each *arg*. Character sequences of the form *%d* in *command*, where *d* is a digit from 1 to 9, are replaced by the *d*'th following unused *arg*. If any such sequences occur, *n* is ignored, and the number of arguments passed to *command* is the maximum value of *d* in *command*.
- ac** Change the % character to the value of *c*.

EXAMPLES

The command line

apply echo *

is similar to **ls(1)**;

apply -2 cmp a1 b1 a2 b2 ...

compares the *a* files to the *b* files;

apply -0 who 1 2 3 4 5

runs **who(1)** five times; and the command

apply 'ln %1 /usr/joe' *

links all files in the current directory to the directory **/usr/joe**.

BUGS

Shell metacharacters in *command* may have bizarre effects; it is best to enclose complicated commands in single quotes `' '`.

You cannot pass a literal `"%2"` if `"%"` is the argument expansion character.

SEE ALSO

sh(1)

NAME

apropos – locate commands by keyword lookup

SYNOPSIS

apropos *keyword* ...

DESCRIPTION

apropos shows which manual sections contain instances of any of the given keywords in their title. It considers each word separately and ignores the case of letters. It considers words that are part of other words; for example, if you use "compile" as a keyword, **apropos** also finds all instances of "compiler".

If **apropos** prints a line that starts with *name(section)* ... you can enter **man section name** to get the documentation for that command.

apropos is actually just the **-k** option to the **man(1)** command.

EXAMPLES

To print a one-line synopsis of each manual section whose table of contents listing contains the keyword "password", enter:

apropos password

To see which manual sections contain the keyword "editor", enter:

apropos editor

When you enter **apropos format**, the **apropos** listing includes the line:

```
printf, fprintf, sprintf (3S)      - formatted output conversion.
```

Enter **man 3s printf** to see the manual entry for the subroutine **printf**.

FILES

/usr/man/whatis Database

SEE ALSO

man(1), **whatis(1)**, **catman(8)**

NAME

ar – archive and library maintainer

SYNOPSIS

ar *key* [*posname*] *afile name* ...

DESCRIPTION

ar maintains groups of files combined into a single archive file. Although its main use is to create and update library files as used by the link editor, you can use **ar** for any similar purpose. The magic string and the file headers that **ar** uses consist of printable ASCII characters. If an archive is composed of printable files, the entire archive is printable.

When **ar** creates an archive, it produces headers in a format that is portable across various machines running UNIX. The portable archive format and structure is described in **ar(5)**. The link editor uses the archive symbol table to effect multiple passes over libraries of object files efficiently. The link editor is further described in **ld(1)**.

ar creates and maintains an archive symbol table and module table only when there is at least one object file in the archive. The archive symbol table is in a specially named file that is always the first file in the archive. This file is never mentioned or accessible. Whenever **ar** creates or updates the contents of such an archive, it also rebuilds the symbol table. Domain/OS BSD **ar** builds a module name table and a long name table in addition to the symbol table; these tables are stored in files that are never mentioned or accessible.

The command key is a required part of **ar**'s command line. *key* is an optional dash (–) followed by one character from the **drqtpmx** set, optionally concatenated with one of more characters from the **vuaibcls** set. The **dtmx** keys can be used with the BSD key argument **A** to act on module names. *posname* is the name of an optional positioning character. *afile* is the archive file. The *names* are constituent files in the archive file.

KEY CHARACTERS

- d** Delete named files from the archive file.
- r** Replace named files in the archive file. If you use the optional character **u** with **r**, only those files with modification dates later than the archive files are replaced. If you use an optional positioning character from the set **abi**, the *posname* argument must be present and specifies that new files are to be placed after (**a**) or before (**b** or **i**) *posname*. Otherwise, new files are placed at the end.
- q** Quickly append named files to the end of the archive file. Optional positioning characters are invalid. Do not check whether the added members are already in the archive. This option is useful for avoiding quadratic behavior when you create a large archive piece by piece. Unchecked, the file can grow exponentially up to the second degree.
- t** Print a table of contents of the archive file. If you do not specify names, table all files. If you specify names, table only those files named.

- p** Print named files in the archive.
- m** Move named files to the end of the archive. If a positioning character is present, the *posname* argument must be present and, as in **r**, specifies where the files are to be moved.
- x** Extract named files. If you do not specify names, all files in the archive are extracted. In neither case does **x** alter the archive file.

KEY ARGUMENTS

- v** Give a file-by-file description of the making of a new archive file from the old archive and the constituent files. When used with **t**, give a long listing of all information about the files. When used with **x**, precede each file with a name.
- c** Create *afile* and suppress the message produced by default when *afile* is created.
- l** Place temporary files in the local (current working) directory rather than in the default temporary directory, */tmp*.
- s** Force the regeneration of the archive symbol table even if you do not invoke **ar(1)** with a command that modifies the archive contents. This option is useful for restoring the archive symbol table after you use the **strip(1)** command on the archive.
- A** Can be used with the keys **dtmx** to respectively delete, print a table of contents, move, or extract by module names.

NOTES

ranlib(1) is not necessary in Domain/OS.

BUGS

If the same file is mentioned twice in an argument list, it may be put in the archive twice.

FILES

*/tmp/v** Temporaries

SEE ALSO

ld(1), **lorder(1)**, **ranlib(1)**, **ar(5)**

NAME

at - execute commands later

SYNOPSIS

at [**-c**] [**-s**] [**-m**] *time* [*day*] [*file*]

DESCRIPTION

The **at** command retains a copy of the named *file* in a spooling area (**/usr/spool/at**), to be used as input to **sh**(1) or **cs**h(1). If you specify the **-c** flag (for **cs**h(1)) or the **-s** flag (for **sh**(1)), **at** uses the specified shell to execute the job; if you don't specify a shell, **at** uses the current environment shell. If you don't specify a filename, **at** prompts for commands from standard input until you type a CTRL/D.

The format of the spool file is as follows: the file contains a four-line header which includes the owner of the job, the name of the job, the shell used to run the job, and indicates whether mail will be sent after the job is executed. The header is followed by a **cd** command to the current directory and a **umask** command to set the modes on any files created by the job. Then **at** copies all relevant environment variables to the spool file. When the script is run, it uses the user and group ID of the spool file's creator.

The *time* argument can be one to four digits, optionally ending in A, P, N, or M for AM, PM, noon, or midnight, respectively. **at** interprets one- and two-digit numbers as hours; three- and four-digit numbers as hours and minutes. If no letters follow the digits, **at** assumes a 24-hour clock.

The optional *day* is either a month name followed by a day number, or a day of the week. If the word **week** follows, invocation is moved seven days further off. You can use recognizable abbreviations for the names of months and days.

at programs are run by periodic execution of the command **/usr/lib/atrun** from **cron**(8). The granularity of **at** depends upon how often **atrun** is executed.

Error output is lost unless you redirect it or request the **-m** flag, in which case a copy of the errors is sent to you via **mail**(1).

OPTIONS

- | | |
|-----------|---|
| -c | Use cs h(1) to execute the job. |
| -m | Send mail to the user after the job has been run. If errors occur during execution of the job, send a copy of the error diagnostics to the user. If no errors occur, send a short message informing the user that no errors occurred. |
| -s | Use sh (1) to execute the job. |

EXAMPLES

To run **agenda** at 8 a.m. on January 24, enter the following command:

at 8am jan 24 agenda

To run **agenda** as input to **cs**h at 3:30 p.m. a week from next Friday, and receive mail after the file is executed, enter the following:

at -c -m 1530 fr week agenda

To run **agenda** as input to **sh** at 12:00 noon a week from today, and receive mail after the file is executed, enter the following:

at -s -m 1200n week agenda

BUGS

Due to the granularity of the execution of **/usr/lib/atrun** from **cron**(1), problems may occur if you schedule activities to take place almost exactly 24 hours in the future.

If the system crashes, **at** does not send you mail informing you that the job wasn't completed.

Sometimes old spool files are not removed from the directory **/usr/spool/at/past**. This is usually due to a system crash, and requires that the files be removed by hand.

DIAGNOSTICS

at complains about various syntax errors and times out of range.

FILES

/usr/lib/atrun	Executor; run by cron (8).
/usr/spool/at	Spooling area
/usr/spool/at/yy.ddd.hhhh.*	Job file; activity for year <i>yy</i> , day <i>dd</i> , hour
hhh	
/usr/spool/at/lasttimedone	Last time atrun was run
/usr/spool/at/past	Directory where jobs are executed from

SEE ALSO

atq (1), **atrm** (1), **calendar** (1), **pwd** (1), **sleep** (1), **cron** (8)

NAME

atq – print the queue of jobs waiting to be run

SYNOPSIS

atq [**-c**] [**-n**] [*user* ...]

DESCRIPTION

atq prints the queue of jobs, created with the **at**(1) command, that are waiting to be run at a later date. With no flags, **atq** sorts the queue in the order that the jobs will be executed.

OPTIONS

-c	Sort the queue by the time that the at command was given.
-n	Print only the total number of files that are currently in the queue.
<i>user...</i>	Display only the files that belong to the specified user(s).

FILES

/usr/spool/at	Spooling area
----------------------	---------------

SEE ALSO

at(1), **atrm**(1), **cron**(8)

NAME

atrm – remove jobs spooled by **at**

SYNOPSIS

atrm [**-f**] [**-i**] [**-**] [[*job #* | *user*] ...]

DESCRIPTION

atrm removes jobs that were created with the **at(1)** command.

OPTIONS

- f** Suppress all information regarding the removal of the specified jobs.
- i** Interactively ask if a job should be removed; a response of “y” removes the job.
- Remove all jobs belonging to the person invoking **atrm**.
- job #* ... Attempt to remove only the jobs identified by the specified job numbers.
- user* ... Remove all jobs belonging to the specified user(s). This form of invoking **atrm** is useful only to the super-user.

FILES

/usr/spool/at Spooling area

SEE ALSO

at(1), **atq(1)**, **cron(8)**

NAME

awk – pattern scanning and processing language

SYNOPSIS

awk [**-F***c*] [*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 *prog* 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*.

Files are read in order; if there are no files, the standard input is read. The filename **-** 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 the **FS** variable name (see below). 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 ; conditional ; 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 can be scalars, array elements (denoted *x[i]*) or fields. Initialize variables to the null string. Array subscripts can be any string, not necessarily numeric; this allows for a form of associative memory. Enclose string constants in quotation marks: **"..."**.

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(3S)**).

The built-in function **length** returns the length of its argument taken as a string, or of the whole line if there is no argument. There are also built-in functions **exp**, **log**, **sqrt**, and **int**. **int** 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(3S)** 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**, (see **grep(1)**). 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) or **!** (for does not contain). A conditional is an arithmetic expression, a relational expression, or a Boolean combination of these.

You can use the special patterns **BEGIN** and **END** to capture control before the first input line is read and after the last. **BEGIN** must be the first pattern; **END** the last.

You can use a single character *c* to separate the fields by starting the program with

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

or by using the **-F*c*** 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 the first two fields in opposite order:

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

Add up the first column, print the 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 }
```

BUGS

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.

SEE ALSO

lex(1), sed(1);

A. V. Aho, B. W. Kernighan, P. J. Weinberger, *Awk - a pattern scanning and processing language*

NAME

basename – strip filename affixes

SYNOPSIS

basename *string* [*suffix*]

DESCRIPTION

basename deletes from *string* any prefix ending in a slash (/) and the *suffix*, if present in *string*. It then prints the result on the standard output. You normally use the **basename** command inside substitution marks (` `) in shell procedures.

EXAMPLE

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

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

SEE ALSO

`csh(1)`, `sh(1)`

NAME

bc — arbitrary-precision arithmetic language

SYNOPSIS

bc [**-c**] [**-l** *name*] [*file* ...]

DESCRIPTION

bc is an interactive processor for a language that resembles C, but provides unlimited precision arithmetic. It takes input from any *file* given, then reads the standard input.

bc is actually a preprocessor for **dc**(1), which it usually invokes automatically, unless the **-c** option is used (see below).

OPTIONS

-c Compile only. Send the input of **dc**(1) to standard output.
-l *name* Include */usr/lib/lib.b* as the arbitrary precision math library to be used.

PROGRAM SYNTAX

The syntax for **bc** programs is shown below (*L* = letters a–z, *E* = expression, *S* = statement).

Comments

Enclosed in */** and **/*.

Names

Simple variables: *L*
 Array elements: *L* [*E*]
 The words *ibase*, *obase*, and *scale*.

Other Operands

Arbitrarily long numbers with optional signs and decimal points
 (*E*)
sqrt (*E*)
length (*E*) Number of significant decimal digits
scale (*E*) Number of digits right of decimal point
L (*E* , ... , *E*)

Operators

+ **-** ***** **/** **%** **^** (**%** is remainder; **^** is power)
++ **--** (Prefix and postfix; apply to names)
== **<=** **>=** **!=** **<** **>**
=+ **=-** **=*** **=/** **=%** **=^**

Statements

```

E
{ S ; ... ; S }
if ( E ) S
while ( E ) S
for ( E ; E ; E ) S
null statement
break
quit

```

Function Definitions

```

define L ( L , ..., L ) {
    auto L , ... , L
    S ; ... S
    return ( E )
}

```

Functions in Math Library

```

s(x)    sine
c(x)    cosine
e(x)    exponential
l(x)    log
a(x)    arctangent
J(n,x)  Bessel function

```

All function arguments are passed by value.

The value of a statement that is an expression is printed unless the main operator is an assignment. Either semicolons or newlines may separate statements. Assignment to **scale** influences the number of digits to be retained on arithmetic operations in the manner of **dc(1)**. Assignments to **ibase** or **obase** set the input and output number radix respectively.

The same letter may be used as an array, a function, and a simple variable simultaneously. All variables are global to the program. "Auto" variables are pushed down during function calls. When using arrays as function arguments or defining them as automatic variables, empty square brackets must follow the array name.

EXAMPLE

The following example:

```
scale = 20
define e(x){
    auto a, b, c, i, s
    a = 1
    b = 1
    s = 1
    for(i=1; i<=x; i++){
        a = a*x
        b = b*i
        c = a/b
        if(c == 0) return(s)
        s = s+c
    }
}
```

defines a function to compute an approximate value of the exponential function and

```
for(i=1; i<=10; i++) e(i)
```

prints approximate values of the exponential function for the first ten integers.

BUGS

The double ampersand (&&) and double pipe (||) characters are not yet valid operators for use with **bc**.

A **for** statement must have all three *E*'s.

A **quit** is interpreted when read, not when executed.

FILES

/usr/lib/lib.b	Mathematical library
/usr/bin/dc	Desk calculator proper

SEE ALSO

dc(1)

NAME

bdf2snf - BDF to SNF font compiler for X11

SYNOPSIS

bdf2snf [-p#] [-s#] [-m] [-l] [-M] [-L] [-w] [-W] [-t] [-i] [bdf-file]

DESCRIPTION

bdf2snf reads a Bitmap Distribution Format (BDF) font from the specified file (or from standard input if no file is specified) and writes an X11 server normal font (SNF) to standard output.

OPTIONS

- p# Force the glyph padding to a specific number. The legal values are 1, 2, 4, and 8. -s# Force the scanline unit padding to a specific number. The legal values are 1, 2, and 4.
- m Force the bit order to most significant bit first.
- l Force the bit order to least significant bit first.
- M Force the byte order to most significant bit first.
- L Force the byte order to least significant bit first.
- w Print warnings if the character bitmaps have bits set to one outside of their defined widths.
- W Print warnings for characters with an encoding of -1; the default is to silently ignore such characters.
- t Expand glyphs in "terminal-emulator" fonts to fill the bounding box.
- i Don't compute correct ink metrics for "terminal-emulator" fonts.

SEE ALSO

X(1), Xserver(1)
 "Bitmap Distribution Format 2.1"

NAME

bdiff – big diff

SYNOPSIS

bdiff *file1 file2* [*n*] [**-s**]

DESCRIPTION

bdiff is used in a manner analogous to **diff**(1) to find which lines in two files must be changed to bring the files into agreement. Its purpose is to allow processing of files which are too large for **diff**.

OPTIONS

- file1* (*file2*) The name of a file to be used. If *file1* (*file2*) is **-**, the standard input is read.
- n* The number of line segments. The value of *n* is 3500 by default. If the optional third argument is given and it is numeric, it is used as the value for *n*. This is useful in those cases in which 3500-line segments are too large for **diff**, causing it to fail.
- s** Specifies that no diagnostics are to be printed by **bdiff** (silent option). Note, however, that this does not suppress possible diagnostic messages from **diff**(1), which **bdiff** calls.

bdiff ignores lines common to the beginning of both files, splits the remainder of each file into *n*-line segments, and invokes **diff** upon corresponding segments. If both optional arguments are specified, they must appear in the order indicated above.

The output of **bdiff** is exactly that of **diff**, with line numbers adjusted to account for the segmenting of the files (that is, to make it look as if the files had been processed whole). Note that because of the segmenting of the files, **bdiff** does not necessarily find a smallest sufficient set of file differences.

FILES

/tmp/bd????

DIAGNOSTICS

Use **help**(1) for explanations.

SEE ALSO

diff(1), **help**(1)

NAME

biff – be notified if mail arrives and who it is from

SYNOPSIS

biff [*y*] [*n*]

DESCRIPTION

biff informs the system whether you want to be notified when mail arrives during the current login session. The command

biff y

enables notification; the command

biff n

disables it.

When mail notification is enabled, the header and first few lines of the message are printed on your screen whenever mail arrives. A **biff y** command is often included in the file `~/.login` or `~/.profile` to be executed at each login.

biff operates asynchronously. For synchronous notification use the MAIL variable of `sh(1)` or the `mail` variable of `csh(1)`.

NOTES

biff only notifies you of mail delivered locally with `/bin/mail` (which includes `mail(1)`, `sendmail(8)`, etc.).

SEE ALSO

`csh(1)`, `sh(1)`, `mail(1)`, `binmail(1)`, `comsat(8C)`

NAME

binmail – send or receive mail among users

SYNOPSIS

```
/bin/mail [ + ] [ -i ] [ person ... ]
/bin/mail [ + ] [ -i ] -f file
```

DESCRIPTION

Note: This is the old version 7 UNIX system mail program. The default **mail** command is described in **mail(1)**, and its binary is in the directory **/usr/ucb**.

With no argument, **mail** prints a user's mail, message-by-message, in last-in, first-out order; the optional argument **+** displays the mail messages in first-in, first-out order. For each message, **mail** reads a line from the standard input to direct disposition of the message. When you log in, **mail** informs you if you have mail.

COMMANDS

<i>newline</i>	Go on to the next message.
d	Delete the message and go on to the next.
p	Print the message again.
-	Go back to the previous message.
s [<i>file</i>] ...	Save the message in the named <i>files</i> (mbox is the default).
w [<i>file</i>] ...	Save the message, without a header, in the named <i>files</i> (mbox is the default).
m [<i>person</i>] ...	Mail the message to the named <i>persons</i> (yourself is the default).
EOT (CTRL/D)	Put unexamined mail back in the mailbox and stop.
q	Same as EOT.
!command	Escape to the shell to do <i>command</i> .
*	Print a command summary.

An interrupt normally terminates the **mail** command; the mail file is unchanged.

OPTIONS

+	Display the mail messages in a first-in, first-out order.
-i	Tells mail to continue after interrupts.
<i>persons</i>	When <i>persons</i> are named, mail takes the standard input up to an end-of-file (or a line with just .) and adds it to each <i>person's</i> "mail" file. The message is preceded by the sender's name and a postmark. Lines that look like postmarks have a > prefix. A <i>person</i> is usually a username recognized by login(1) . Prefix the system name and exclamation mark to <i>person</i> to denote a recipient on a remote system (see uucp(1C)).

-f file The **-f** option causes the named file, for example, **mbox**, to be printed as if it were the mail file.

BUGS

Race conditions sometimes result in a failure to remove a lock file.
Normally anybody can read your mail. An installation can overcome this by making **mail** a set-user-id command that owns the mail directory.

FILES

/etc/passwd	To identify sender and locate per-
sons	
/usr/spool/mail/*	Incoming mail for user
~/mbox	Saved mail
/tmp/ma*	Temp file
/usr/spool/mail/*.lock	Lock for mail directory
~/dead.letter	Unmailable text

SEE ALSO

mail(1), write(1), uucp(1C), uux(1C), sendmail(8)

NAME

bldt – display time operating system was built

SYNOPSIS

bldt [*options*] [*node_id*]

DESCRIPTION

bldt displays the time at which the running version of Domain/OS was built.

node_id (optional) Display the build time of the node whose network root directory is *pathname*.

Default if omitted: display build time of current node

OPTIONS

-n *node_spec* ... Display build time of specified node[s].

-a Display build time of all nodes.

EXAMPLES

\$ bldt //ward

```
**** Node 29C27.4B51 ****   "//ward"
Domain/OS kernel(3), revision 10.0, bl20.1 \
                                April 15, 1988  1:02:54 pm
```

\$ bldt -n //june

```
**** Node 29C27.CBB9 ****   "//june"
Domain/OS kernel(8), revision 10.0, bl17.3 \
                                February 9, 1988  8:12:37 am
```

\$ bldt -n CBB9

```
**** Node 29C27.CBB9 ****   "//june"
Domain/OS kernel(8), revision 10.0, bl17.3 \
                                February 9, 1988  8:12:37 am
```

NAME

cal – print calendar

SYNOPSIS

cal [*month*] *year*

DESCRIPTION

cal prints a calendar for a specified year. If you specify a month, **cal** prints a calendar just for that month. The range for *year* is 1-9999; for *month*, 1-12. Both *year* and *month* must be Arabic numbers. **cal** produces the Gregorian calendar for dates after 1752 (when this calendar was adopted by England and her colonies); **cal** produces the Julian calendar for dates before 1752.

EXAMPLES

Try:

cal 9 1752

To print a calendar for the entire year of 1989, type the following:

cal 1989

To print a calendar for December, 1985 only, use this command:

cal 12 1985

BUGS

Note that specifying **cal 89** prints a calendar for the year 89, not 1989.

cal always assumes the year to start in January, even though this is historically naive.

NAME

calendar – reminder service

SYNOPSIS

calendar [-]

DESCRIPTION

calendar provides an individual reminder service by consulting the file **calendar** in the current directory and printing out lines that contain today's or tomorrow's date anywhere in the line. You must create the file before **calendar** can successfully run.

calendar recognizes most month-day entries, such as Dec. 7, december 7, 12/7. It does not recognize day-month entries, such as 7 December or 7/12. If you give the month as an asterisk (*) with a date, for example, * 1, that day in any month will do. On weekends "tomorrow" extends through Monday.

When you specify an argument, **calendar** does its job for all users who have a file named **calendar** in their login directories and sends any positive results by mail. Normally **calendar** does this early every day, under the control of **cron**(8).

The file **calendar** is first run through the C preprocessor, **/lib/cpp**, to include any other calendar files specified with the usual **#include** syntax. Included calendars are usually shared by all users, and maintained and documented by the local administration.

EXAMPLE

A typical line in your **calendar** file may look like this:

12/15 Departmental meeting at 3 p.m.

BUGS

calendar's extended idea of "tomorrow" doesn't account for holidays.

FILES

/calendar	
/usr/lib/calendar	To figure out today's and tomorrow's dates
/etc/passwd	
/tmp/cal*	
/lib/cpp, egrep, sed, mail	As subprocesses

SEE ALSO

at(1), **cron**(8), **mail**(1)

NAME

cat – catenate and print

SYNOPSIS

cat [**-u**] [**-n**] [**-s**] [**-v**] *file* ...

DESCRIPTION

The **cat** command reads each *file* in sequence and displays it on the standard output. Thus,

cat file

displays the file on the standard output, and

cat file1 file2 >file3

concatenates the first two files and places the result on the third.

If no input file is specified, or if the argument **-** is encountered, **cat** reads from the standard input file. Output is buffered in the block size recommended by **stat(2)**, unless the standard output is a terminal, when it is line buffered.

OPTIONS

- u** Make the output completely unbuffered.
- n** Display the output lines preceded by line numbers, numbered sequentially from one. You can specify **-b** with **-n**, if you want to omit the line numbers from blank lines.
- s** Crush out multiple adjacent empty lines so that the output is displayed single-spaced.
- v** Display non-printing characters. Control characters appear like **^X** for CTRL/X; the delete character (octal 0177) appears as **^?**. Non-ASCII characters (with the high bit set) are printed as **M-** (for meta) followed by the character of the low 7 bits. If you specify the **-e** option with the **-v** option, display a dollar sign (\$) character at the end of each line. If you specify the **-t** option with the **-v** option, display tab characters as **^I**.

BUGS

Beware of **cat a b >a** and **cat a b >b**, which destroy the input files before reading them.

SEE ALSO

cp(1), **ex(1)**, **more(1)**, **pr(1)**, **tail(1)**

NAME

cb – C program beautifier

SYNOPSIS

cb

DESCRIPTION

cb places a copy of the C program from the standard input on the standard output, with spacing and indentation that display the structure of the program.

SEE ALSO

indent(1)

NAME

cc - C compiler

SYNOPSIS

cc [*options*] *file* ...

DESCRIPTION

cc is the Domain/OS BSD interface to the preprocessor (**cpp**), the Domain C compiler, and the link editor (**ld**). **cc** accepts several types of arguments. It interprets arguments with names ending in **.c** as C source programs, compiles them, and leaves each object program on the file whose name is that of the source, with **.o** substituted for **.c**. However, **cc** normally deletes the **.o** file if a single C program is compiled and loaded all at one go. Assembly source programs (arguments whose names end with **.s**) are not supported.

You can typically improve run-time performance if you compile your code using the **-A cpu** option with a processor-specific id. Use an argument appropriate to the CPU type of the machine(s) on which you will run the executable file. The option **-A cpu 3000**, for example, generates optimal code for the DSP90, DN330, DN560, DN570, DN580, DN3000, DN3500, DN4000, and DN4500 processors.

Not all standard UNIX options are available. The Domain/OS BSD **cc** command provides some unique options. The **cc** command will interpret any command line argument beginning with a dash (**-**), as a compiler or a preprocessor option. If **cc** doesn't recognize an option, it assumes that it is an option for the link editor **ld(1)** and passes it along. **cc** recognizes the following as preprocessor options: **-C**, **-D**, **-H**, **-I**, and **-U**. The following are link editor options: **-a**, **-l**, **-m**, **-o**, **-r**, **-s**, **-t**, **-u**, **-x**, **-z**, **-L**, **-M**, **-V**, and **-A**.

OPTIONS

- a** Produce an object file for execution. This is the default. Use **-r** to retain relocation information in the object module. If you specify both **-a** and **-r**, the link editor retains relocation information for all data except common symbols, which it allocates. (**ld** switch)
- Bstring** Prefix pathname to **cpp**, **cc**, and **ld** for substitute preprocessor, compiler, and linker passes. If name is empty it is taken to be **/usr/lib/o**.
- c** Suppress the linking phase of the compilation and force an object file to be produced, even if only one program is compiled.
- C** Prevent the preprocessor from stripping comments. (**cpp** switch)
- Dname=def**
- Dname** Define the *name* to the preprocessor, as if by **#define**. If you do not specify a definition, define *name* as 1. The **-D** switch has lower precedence than **-U**; if you use both for the same *name*, the *name* is undefined, regardless of the order the switches appear in. (**cpp** switch)

- E** Run only the macro preprocessor on the named C programs, and send the result to the standard output.
- g** Generate full run-time debugger information. This is the same as the Aegis **-dbs** option to the Domain C compiler.
- H** Print to **stderr** the pathname of every file included during this compilation. (**cpp** switch)
- Idir** Change the search path for **#include** files with names not beginning with a slash (/). Look first in the directory of the source file; then in directories named in this option; and finally, in directories on a standard list.
- lx** Search the library named **libx.a**. **cc** searches libraries in the order that they appear on the command line. By default, the link editor searches for libraries first in the directory **/lib**, then in **/usr/lib**. You can specify additional library directories with the **-L** switch. (**ld** switch)
- Ldir** Change the search path for libraries. By default, the compiler looks for **libx.a** libraries in the directories specified by **/lib** and **/usr/lib**. This option allows you to specify a different directory before searching these standard directories. This is useful if you have different versions of a library and you want to specify which one the link editor should use. Note that this switch is effective only if it precedes a **-l** switch. (**ld** switch)
- m** Produce a map or listing of the input/output sections on standard input. (**ld** switch)
- M** Run only the macro preprocessor on the named C programs, requesting it to generate *Makefile* dependencies and send the result to the standard output.
- o output** Name the final output file *output*. By default, *output* is **a.out**. If you specify a different name, the system leaves any existing **a.out** file undisturbed.
- O** Turn on compiler optimizations.
- p** Arrange for the compiler to produce code which counts the number of times each routine is called. If the command does load, replace the standard startup routine by one which automatically calls **monitor(1)** at the start and arranges to write out a **mon.out** file at normal termination of execution of the object program. An execution profile can then be generated by use of **prof(1)**.
- P** Run only the macro preprocessor on the named C program, and leave the result on corresponding files suffixed with **.i**.
- pg** Causes the compiler to produce counting code in the manner of **-p**, but invokes a run-time recording mechanism that keeps more

extensive statistics and produces a **gmon.out** file at normal termination. An execution profile can then be generated by use of **gprof(1)**.

- r** Retain relocation entries in the output object module. Relocation entries must be preserved if the object file will be specified in a future **ld** or **bind** command. The default is **-a**. (**ld** switch)
- s** Strip line-number entries and symbol-table information from the output object file. The switch is equivalent to using the **strip(1)** utility and is useful if you want to reduce the size of the object module. Note, however, that removing this information from a program makes it impossible to debug the program with a source level debugger (**dbx** or **dde**). (**ld** switch)
- t[p0l]** Find only the designated preprocessor (**p**), compiler passes (**0**), or linker (**l**) in the files whose names are constructed by a **-B** option. In the absence of a **-B** option, the name is taken to be **/usr/lib/n**. The value **-t ""** is equivalent to **-t0l**. The **-Y** option performs the same function and is easier to use.
- Tsystype** Define the target system type (*systype*) for the compiled object; see the **-A sys[type],sys** option for a list of valid *systype* values.
- u symname** Enters *symname* as an undefined symbol in the symbol table. This switch is useful if you are using the **cc** command to load a library. The symbol table is initially empty and needs an unresolved reference to force **ld** to load the first routine. (**ld** flag)
- Uname** Remove any initial definition of *name*. (**c++** switch)
- V** Output a message giving information about the version of **ld** being used. (**ld** switch)
- w** Suppress warning diagnostics.
- Wc,arg1[,arg2]** Hand off the arguments *argi* to pass *c* where *c* is one of **p**, **0**, or **l**, indicating the preprocessor, compiler, or linker. Using **-W0** enables you to use */com/cc options* that are not available with */bin/cc*. For example: **-W0,-opt,1** passes **-opt 1** to the compiler.

The following */com/cc options* can be used in this manner with the **-W0** option:

Default options are indicated by "(D)".

- ac** (D) Produce absolute code. The alternative option is **-pic**, which forces the compiler to produce position-independent code.
- alnchk** (D) Display messages about alignment of structures.

-nalnchk	Suppress alignment messages.
-nansi	(No ANSI) Do not use any ANSI features.
-bss	Put uninitialized global variables in the .bss section of the object file.
-nbss (D)	Put uninitialized global variables in named sections.
-comchk	Check for balanced comments and warn if comments are not balanced.
-ncomchk (D)	Suppress comment checking.
-compress	Store object file data in compressed form. If neither -compress nor -ncompress is specified, the compiler will compress all data sections except the one called ".data".
-ncompress (D)	Store object file data in uncompressed form. If neither -compress nor -ncompress is specified, the compiler compresses all data sections except the one called ".data".
-cond	Compile lines prefixed with "#debug".
-ncond (D)	Ignore lines prefixed with "#debug".
-exp	Generate expanded code listing (implies -l).
-nexp (D)	Suppress expanded code listing.
-frnd	Force the compiler to write all floating-point operands to memory and then fetch the memory contents before evaluating the expression. This ensures that each operand will have the same amount of precision so that floating-point comparisons will produce correct results. If you do not compile with -frnd , floating-point operands may be kept in registers, which support more accuracy than memory. Consequently, when a register operand is compared with a memory

	operand, the result may not be what is expected. This is particularly true of equality comparisons.
-indexl	Use 32-bit indexing for all array references.
-nindexl (D)	Use source code's array dimension information to determine whether to use 16-bit or 32-bit indexing.
-info level	Control the output of informational messages. The four informational levels are 0, 1, 2 and 3. -info 0 is the default; it suppresses informational messages.
-llib [pathname]	Specify one or more libraries that are not currently installed but should be installed when the program is executed. These libraries are searched at compile time to determine whether indirect or absolute references should be generated.
-l [file]	Generate compilation listing. If no pathname is specified, the listing is written to a file with the same name as the source file, but with the suffix .lst replacing .c.
-nl (D)	Suppress compilation listing.
-map	Generate symbol table map (implies -l).
-nmap (D)	Suppress symbol table map.
-mgb1 (D)	Map global names on output to maintain case sensitivity.
-nmgb1	Do not map global names; output all names in uppercase.
-msgs	Force compiler to print final message summarizing listing number of errors and warnings.
-nmsgs (D)	Suppresses final compilation message.
-natural	Use natural alignment for structures and array elements that do not have alignment attributes.

-nnatural (D)	Suppress natural alignment for structures and array elements that do not have alignment attributes.
-nclines	Suppress the generation of COFF line number tables.
-opt [<i>n</i>] (D)	Perform global optimization. 'n' is a single-digit integer in the range 0 to 4, indicating the level of optimization performed. -opt 0 is identical to the obsolete switch -nopt . If 'n' is omitted, 3 is the default level. Increasing the optimization level results in faster execution of your program, at the expense of longer compilation time. See the <i>Domain C Language Reference</i> manual for details.
-pic	Produce position-independent object code. The default is to produce absolute code.
-prasm (D)	Use Series 10000 assembly language format for the expanded listing generated by -exp . Note: this option has no effect if used without the -exp option.
-nprasm	Use 68000 assembly language format for the expanded listing generated by -exp . Note: this option has no effect if used without the -exp option.
-prof	Produce code that, when executed, produces a ".mon" file that can be used by the prof utility to evaluate performance.
-std	Flag usage of nonstandard features with warnings.
-nstd (D)	Do not generate warnings for usage of nonstandard features.
-nstdc	Set the value of <code>__STDC__</code> to 0. Use this option along with -xansi if portions of your code use ANSI features not yet implemented by

- Domain C and the unimplemented ANSI features are contingent on `__STDC__` being defined.
- `-type (D)` Obsolete option. Use `-xansi`.
 - `-ntype` Obsolete option. Use `-nansi`.
 - `-uline (D)` Recognize `#line` preprocessor options for source file line numbering. The compiler will report errors in terms of the line numbers set by this option. The debugger line number table will be built with these line numbers.
 - `-nuline` Ignore `#line` preprocessor options.
 - `-xansi` (Extended ANSI) Use all ANSI features currently available; also use all Domain extensions. The ANSI features currently implemented are: function prototyping, `const`, `volatile`, and signed qualifiers, `__STDC__` macro defined as 1.
- `-Y[p0ISILU], dir` Specify a new pathname and directory for the locations of the tools and directories designated by the first argument. You can include only one letter or number per `-Y` switch, but there is no limit to the number of `-Y` switches per compilation. The valid letters and numbers, and their meanings, are as follows:
- | | |
|----------|--|
| P | Preprocessor (cpp) |
| 0 | Compiler (cc) |
| I | Link editor (ld) |
| S | Directory containing the startup routine (/usr/lib/crt0.o) |
| I | Default include directory searched by the preprocessor (/usr/include) |
| L | First default library directory searched by the link editor |
| U | Second default library searched by the link editor |
- If the location of a tool is being specified, the new pathname for the tool is `/dir/tool`. If more than one `-Y` option is applied to any one tool or directory, the last occurrence holds.

The Domain/OS BSD version of `cc` does not support the following options: `-go`, `-lg`, `-R`, `-S`, `-f`.

Domain/OS BSD EXTENSIONS

The `-A` option identifies Domain/OS BSD extensions to `cc` and `ld`; see `ld(1)` for additional information about this option.

- A nansi** Do not compile with ANSI rules. Same as **-nansi /com/cc** option.
- A cpu,*id*** Generate code for a particular class of processor; the default *id* is **any**. Values for *id* are as follows:

any	Series 10000 code, if you're compiling on a Series 10000 workstation, or 680x0 code, if you're compiling on a 680x0-based workstation
a88k	Series 10000 code
m68k	680x0-based code
160	DSP160 code
460	DSP460 code
660	660 code
90	DSP90 code
330	DN330 code
560	DN560 code
570	DN570 code
580	DN580 code
3000	DN3000 code and DN4000 code
FPA1	Floating-Point Accelerator
FPX	Floating-Point Accelerator Board
PEB	Performance Enhancement Board

The following values for *id* are synonyms: 160, 460, and 660. Also, the following values for *id* are synonyms: 90, 330, 560, 570, 580, and 3000. Using a processor-specific code generation mode can result in programs that run faster.

- A sys[*type*],*sys*** Define the target system type (*sys*) for the compiled object; *sys* may be one of the following:

any	Version independent
bsd4.2	Berkeley version 4.2
bsd4.3	Berkeley version 4.3
sys5	UNIX System V
sys5.3	UNIX System V Release 3

This option replaces the **-Tsystype** option (which is supplied for backwards compatibility).

- A run[*type*],*sys*** Like **-A sys[*type*],*sys*** this option passes runtime information to the compiler and linker.

Other arguments are taken to be loader option arguments, or C-compatible object programs, typically produced by an earlier **cc** run, or perhaps libraries of C-compatible routines. These programs, together with the results of any compilations specified, are loaded (in the order given) to produce an executable program with name **a.out**.

EXAMPLE

The following example compiles **filename.c** using the compiler in **/usr/lib/old/cc**

```
cc -Y0/usr/lib/old filename.c
```

DIAGNOSTICS

The diagnostics produced by C itself are intended to be self-explanatory. The loader may produce occasional messages.

FILES

<i>file.c</i>	Input file
<i>file.o</i>	Object file
<i>a.out</i>	Loaded output (default)
<i>/tmp/ctm?</i>	Temporary
<i>/usr/lib/cpp</i>	Preprocessor
<i>/usr/apollo/lib/cc</i>	Compiler
<i>/bin/ld</i>	Link editor
<i>/usr/lib/crt0.o</i>	Runtime startoff
<i>/usr/lib/mcrt0.o</i>	Startoff for profiling
<i>/usr/lib/gcrt0.o</i>	Startoff for gprof-profiling
<i>/usr/include</i>	Standard directory for #include files
<i>mon.out</i>	File produced for analysis by prof(1)
<i>gmon.out</i>	File produced for analysis by gprof(1)

The following files are not supported:

<i>/lib/ccom</i>	Compiler
<i>/lib/sccom</i>	Compiler for single precision floats
<i>/usr/c/ocom</i>	Backup compiler
<i>/usr/c/ocpp</i>	Backup preprocessor
<i>/lib/c2</i>	Optional optimizer
<i>/lib/libc.a</i>	Standard library
<i>/usr/lib/libc_p.a</i>	Profiling library

SEE ALSO

dbx(1), **dde(1)**, **gprof(1)**, **ld(1)**, **prof(1)**, **monitor(3)**

Domain C Language Reference

Domain/OS Programming Environment Reference

B. W. Kernighan and D. M. Ritchie, *The C Programming Language*, Prentice-Hall, 1978

B. W. Kernighan, *Programming in C—a tutorial*

D. M. Ritchie, *C Reference Manual*

NAME

cd – change working directory

SYNOPSIS

cd *directory*

DESCRIPTION

cd changes the current working directory to *directory*. The process must have execute (search) permission in *directory*.

Because a new process is created to execute each command, **cd** would be ineffective if it were written as a normal command. Therefore, it is an internal command in the Bourne shell, Korn shell, and the C shell.

When using **csh**(1) you may specify a list of directories in which *directory* is to be sought as a subdirectory, if it is not a subdirectory of the current directory. See the description of the **cdpath** variable in **csh**(1).

SEE ALSO

csh(1), **sh**(1), **ksh**(1), **pwd**(1), **chdir**(2)

NAME

cdc - change the delta commentary of an SCCS delta

SYNOPSIS

cdc *-rSID* [*-m[mrlist]*] [*-y[comment]*] *file ...*

DESCRIPTION

cdc changes the delta commentary for a specific SID of each named *file*. Delta commentary is defined to be the modification request (MR) and comment information normally specified via the *-m* and *-y* arguments to the **delta**(1) command.

If you specify a *directory*, **cdc** behaves as though each file in the directory is specified as a named file, except that it silently ignores non-SCCS and unreadable files. If you specify a name of *-*, **cdc** reads the standard input. It takes each line of the standard input to be the name of an SCCS file to be processed.

Arguments to **cdc**, which may appear in any order, consist of *options* and filenames. All options described below apply independently to each named file.

OPTIONS

-rSID

Specify the SCCS identification string (SID) of a delta for which the delta commentary is to be changed.

-m[mrlist]

Supply a list of MR numbers to be added and/or deleted in the delta commentary of the SID specified by the *-r* option. The SCCS file must have the *v* flag set. A null MR list has no effect.

MR entries are added to the list of MRs in the same manner as that of **delta**(1). To delete an MR, precede the MR number with an exclamation point (!). If the MR to be deleted exists in the current list of MRs, **cdc** removes it and changes it into a "comment" line. **cdc** places a list of all deleted MRs in the comment section of the delta commentary, preceding the list by a comment line stating that the MRs were deleted.

If you do not specify *-m*, and the standard input is a terminal, **cdc** issues the prompt "MRs?" on the standard output before it reads the standard input. If the standard input is not a terminal, **cdc** does not issue a prompt. The "MRs?" prompt always precedes the "comments?" prompt (see the *-y* option).

MRs in a list are separated by blanks and/or tab characters. An unescaped newline character terminates the MR list.

Note that if the `v` flag has a value, it is taken to be the name of a program (or shell procedure) that validates the correctness of the MR numbers. If a nonzero exit status is returned from the MR number validation program, `cdc` terminates and the delta commentary remains unchanged.

`-y[comment]`

Supersede the existing "comment(s)" for the delta specified by the `-r` option. The previous comments are kept, but preceded by a comment line stating that they were changed. A null "comment" has no effect. If you do not specify `-y`, and the standard input is a terminal, `cdc` issues the prompt "comments?" on the standard output before it reads the standard input. If the standard input is not a terminal, `cdc` does not issue a prompt. An unescaped newline character terminates the "comment" text.

EXAMPLE

To add `bl78-12345` and `bl79-00001` to the MR list, remove `bl77-54321`, and add the comment "trouble" to delta 1.6 of `s.file`:

```
cdc -r1.6 -m"bl78-12345 !bl77-54321 bl79-00001" -ytrouble s.file
```

BUGS

If you supply SCCS filenames to `cdc` via the standard input (`-` on the command line), you must also use the `-m` and `-y` options.

To modify the delta commentary, you must be either the creator of the delta or the owner of the SCCS file and directory.

FILES

x-file	See <code>delta(1)</code> .
z-file	See <code>delta(1)</code> .

DIAGNOSTICS

Use `help(1)` for explanations.

SEE ALSO

`admin(1)`, `delta(1)`, `get(1)`, `help(1)`, `prs(1)`

NAME

chacl – change access control list

SYNOPSIS

```
chacl [ -odfvLR ] <spec> file...
chacl [ -odfvLR ] -D <sid> file ...
chacl [ -odfvLR ] [ -u <owner> ] [ -g <group> ] [ -z <organization> ] file...
chacl [ -odfvLR ] { -c | -l | -n } file ...
chacl [ -vLR ] { -B | -S } file...
```

DESCRIPTION

The **chacl** command changes the entries in an object's access control list (ACL). Use the specification (*spec*) part of the command line either to set the rights for a given subject identifier (*sid*), or to change the inheritance mechanisms of a directory. The specification syntax, shown below, is similar to **chmod**'s symbolic mode form.

```
<spec>:      <sid><op><rights> | <req><op><inh> | <spec>[,<spec>...]
<sid>:       %.%.% | <req>
<req>:       [ugzo] | a
<op>:        = | + | -
<inh>:       [UP]
<rights>:    [prwxksl]
```

OPTIONS

- B
-S The -B (BSD) and -S (SysV) options simply set a directory to use the appropriate semantics. Any existing ACLs are removed, and the protections on the directory are determined by the current **umask**(2). Owner, group, and organization inheritance are determined using the appropriate semantics (SysV, all from current process; BSD, owner from current process, group from directory. Organization is marked "ignore" for both).
- c Force calculation of the extended entry mask. The mask represents the maximum rights of all extended ACL entries, and is automatically calculated each time **chacl** is run. This option is used to undo the effects of the **chmod** command, as **chmod** affects the mask as well as the world required entry (%.%.%.) when changing rights for "other".
- l Set local access. With local access set, an object can be accessed only from the node on which it is located.
- n Set network access.
- o Make the changes on the ACL itself for the objects specified. If the -o, -d, or -f options are not specified, -o is assumed. These options can be used in any combination.
- d Make the changes on the initial directory ACL.

- f** Apply the changes to the initial file ACL.
- v** (verbose) List each destination as the ACL is changed.
- L** Follow any soft links encountered, and operate on the object to which the link points. Since soft links in Domain/OS do not have ACLs, attempting to change a soft link without the **-L** flag simply results in a warning, with no change.
- R** Apply the changes recursively to any directories encountered among the files listed. Be very careful when combining this option with the **-L** option!
- D** Delete extended entries from an ACL. Required entries may not be deleted, so *<sid>* must be an actual subject identifier (see below).
- u** Set the owner field in an ACL.
- g** Set the group field in an ACL.
- z** Set the organization field in an ACL.

SUBJECT IDENTIFIERS

The *sid* (Subject Identifier) used in the first form (*<sid><op><rights>*) is a way of specifying a user or set of users. It may include a username, group name and organization name, any of which may be replaced with the wildcard **%**, or left off, as described in *acl*(7). The special cases **u**, **g**, **z** and **o** refer to the required entries in the ACL for user, group, organization and world. The special case **a** refers to the all of the above (user, group, organization and world), as does a null SID field. These special cases do not affect required entries that are marked "Ignore". Short user IDs that are a combination of the letters **u**, **g**, **z** and **o** are distinguished from the special cases by the use of the **%** syntax described above. Thus **oz+x** adds execute rights for other and organization, whereas **oz.%.%+x** or **oz..+x** adds execute rights for just the user **oz**.

ACCESS RIGHTS

Access rights are specified by the *op* (operator) and *rights* parameters to **chacl**. Valid operators are **=**, **+**, and **-**. The **=** operator specifies absolute rights for the SID. If the ACL already contains an entry for this SID, **acl** changes it to contain the rights listed. Otherwise, it adds an entry with the specified SID and access rights.

If you specify the **+** operator, the rights are added to any existing rights for the specified SID. Likewise, the **-** operator removes the rights from the ACL entry for the SID. If no entry exists for the SID, the entire ACL is searched for more general entries that apply to this SID. The specified rights are then added to or removed from this aggregate set of rights, and a new entry is created for the specific SID.

Access rights consist of any combination of the following letters:

Files:

- p** Protect rights; allow rights to be changed
- r** Read rights; allows file to be read
- w** Write rights; allows file to be written
- x** Execute rights; allows file to be executed
- k** Keep; prevents file from being deleted or having its name changed
- s** Set ID; usable only with **u**, **g** and **z** (user, group, and organization); causes this executable to be run with the effective ID of the user, group or organization

Directories:

- p** Protect rights; allow rights to be changed
- r** Read rights; allows directory to be listed
- w** Write rights; allows names to be added, changed or deleted
- x** Execute rights; allows subordinate objects to be used, without allowing the directory to be listed; also called search rights
- k** Keep; prevents directory from being deleted or having its name changed

The following is used alone, and overrides any other rights specified:

- I** Ignore; used to ignore the rights in the required owner, group, organization, and other entries

To change the inheritance properties of a directory, use the second form of ACL specification (*<req><op><inh>*). In this case, the first field must consist only of required entries **u**, **g**, **z**, or **a** (user, group, organization, all) and the second field specifies the inheritance option.

The valid inheritance options are as follows:

- U** Umask; the rights for new objects in this directory are those requested by the process creating them as modified by the **umask(2)** of that process
- P** Process; inherit user, group or organization from the process creating a new object in this directory

EXAMPLES

```
chacl g+w *
```

Add write rights for the group to each file in this directory.

```
chacl a=rx foo
```

Give owner, group, organization and world read and execute rights to the file **foo**.

```
chacl ugz=I .
```

Ignore the required entries for owner, group and organization.

```
chacl %.os=prwx .
```

Give the **os** group full rights to this directory.

```
chacl ..mktg-pw,..r_d=prwx .
```

Be sure that the **mktg** organization does not have write or protect rights and that **r_d** has full rights to the current directory.

```
chacl -B /usr/u/bar
```

Strip any extended ACLs from **/usr/u/bar**, and set it up as a BSD directory.

```
chacl -D arnold.staff *
```

Delete any ACL entries referring to **arnold.staff**.

```
chacl -od susan+x /usr/u/zap
```

Always allow **susan** to use objects in **/usr/u/zap** directory, and to search any new sub-directories.

```
chacl -odf user= magicdir
```

Insure that **user** has no rights to **magicdir**, nor to any files or sub-directories subsequently created in **magicdir**.

```
chacl -f ugz=UP .
```

Newly created files in the current directory inherit owner, group, and organization (and the associated rights) from the process.

```
chacl -df g-P .
```

Do not inherit group from the process, that is, inherit it from this directory for new files and sub-directories.

SEE ALSO

lsacl(1), cpacl(1), chmod(1), chgrp(1), chorg(1), chown(2), umask(2), acl(7), salacl(8)

NAME

checknr – check nroff/troff files

SYNOPSIS

checknr [**-s**] [**-f**] [**-a** *x1.y1* *x2.y2* ... *xn.yn*] [**-c** *x1* *x2* *x3* ... *xn*] [*file* ...]

DESCRIPTION

checknr checks a list of **nroff**(1) or **troff**(1) input files for certain kinds of errors involving mismatched opening and closing delimiters and unknown commands. If you don't specify any files, **checknr** checks the standard input. The following delimiters are checked:

- Font changes using **\fx** ... **\fP**.
- Size changes using **\sx** ... **\s0**.
- Macros that come in open ... close forms; for example, the **.TS** and **.TE** macros, which must always come in pairs.

checknr knows about the **ms**(7) and **me**(7) macro packages.

checknr is intended for use on documents prepared with **checknr** in mind, much the same as **lint**(1). It expects a certain document writing style for **\f** and **\s** commands. Each **\fx** must be terminated with **\fP** and each **\sx** must be terminated with **\s0**. While you may go directly into the next font, or explicitly specify the original font or point size (many existing documents actually do this), this practice produces complaints from **checknr**. Since it is probably better to use the **\fP** and **\s0** forms anyway, you should think of this as a contribution to your document preparation style.

OPTIONS

- a** Add pairs of macros to the list. This option must be followed by groups of six characters, each group defining a pair of macros. The six characters are a period, the first macro name, another period, and the second macro name. For example, to define a pair **.BS** and **.ES**, use **-a.BS.ES**
- c** Define commands which **checknr** would otherwise complain about as undefined.
- f** Ignore **\f** font changes.
- s** Ignore **\s** size changes.

DIAGNOSTICS

Complaints about unmatched delimiters
 Complaints about unrecognized commands
 Various complaints about the syntax of commands

BUGS

There is no way to define a one-character macro name using **-a**.
checknr does not correctly recognize certain reasonable constructs, such as conditionals.

SEE ALSO

nroff(1), **troff**(1), **checkeq**(1), **ms**(7), **me**(7)

NAME

chgrp – change group

SYNOPSIS

chgrp [**-fR**] *group file ...*

DESCRIPTION

chgrp changes the group ID of the *files* to *group*. The group can be either a decimal GID or a group name found in the group ID file.

To invoke **chgrp**, you must belong to the specified group and be the owner of the file, or be the super-user.

OPTIONS

- | | |
|-----------|--|
| -f | When you specify the -f (force) option, chgrp does not report errors. |
| -R | When you specify the -R option, chgrp recursively descends its directory arguments setting the specified group ID. When chgrp encounters symbolic links, it changes their group but they are not traversed. |

FILES

/etc/group

SEE ALSO

chacl(1), chown(2), passwd(5), group(5), chown(8)

NAME

chmod – change mode

SYNOPSIS

chmod [**-Rf**] *mode file ...*

DESCRIPTION

chmod changes the mode of each named *file* according to *mode*, which may be absolute or symbolic.

An absolute *mode* is an octal number constructed from the OR of the following modes:

4000	Set user ID on execution
2000	Set group ID on execution
1000	Sticky bit (the sticky bit is not supported by Domain/OS BSD)
0400	Read by owner
0200	Write by owner
0100	Execute (search in directory) by owner
0070	Read, write, execute (search) by group
0007	Read, write, execute (search) by others

A symbolic *mode* has the following form:

[*who*] *op permission* [*op permission*]

who is a combination of the letters **u** (for user's permissions), **g** (group), and **o** (other). The letter **a** stands for all, or **ugo**. If you omit *who*, the default is **a**, but the setting of the creation mask is taken into account. See **umask(2)** for further details on this.

op can be **+** to add *permission* to the file's mode, **-** to take away *permission*, or **=** to assign *permission* absolutely (all other bits are reset).

permission is any combination of the letters **r** (read), **w** (write), **x** (execute), **X** (set execute only if file is a directory or some other execute bit is set), **s** (set owner or group ID), and **t** (save text – sticky). The letters **u**, **g**, or **o** indicate that *permission* is to be taken from the current mode. Omitting *permission* is useful only with an equal sign (**=**) to take away all permissions.

You can specify multiple symbolic modes separated by commas. Operations are performed in the order specified. The letter **s** is useful only with **u** or **g**.

Only the owner of a file (or the super-user) can change its mode.

OPTIONS

- R** When you specify **-R**, **chmod** recursively descends its directory arguments, setting the mode for each file as described above. When symbolic links are encountered, their mode is not changed and they are not traversed.
- f** When you specify this option, **chmod** does not complain if it fails to change the mode on a file.

EXAMPLES

To deny others write permission to **file1**, enter the following command:

```
chmod o-w file1
```

To make **file1** executable, enter the following command:

```
chmod +x file1
```

To make **file1** executable by all if it's executable by anyone, enter the following command:

```
chmod +X file1
```

To allow the owner of **file1** read, write, and execute permissions to that file, but allow the group and others only read permission, enter the following command:

```
chmod 744 file1
```

SEE ALSO

chacl(1), ls(1), lsacl(1), chmod(2), stat(2), umask(2), chown(8)

NAME

clear – clear terminal screen

SYNOPSIS

clear

DESCRIPTION

clear clears your terminal screen if this is possible. It looks in the environment for the terminal type, and then looks in **/etc/termcap** to determine how to clear the screen.

FILES

/etc/termcap Terminal capability database

NAME

cmp – compare two files

SYNOPSIS

cmp [-l] [-s] *file1 file2*

DESCRIPTION

cmp compares two specified files. If you specify a dash (-) in place of *file1*, **cmp** uses the standard input. Under default options, **cmp** makes no comment if the files are the same; if they differ, it announces the byte and line number at which the difference occurs. If one file is an initial subsequence of the other, **cmp** notes that fact.

OPTIONS

- l Print the byte number (decimal) and the differing bytes (octal) for each difference.
- s Print nothing for differing files; return codes only.

DIAGNOSTICS

Exit code 0 is returned for identical files; 1 for different files; 2 for an inaccessible or missing argument.

SEE ALSO

comm (1), diff (1)

NAME

coffdump – dump selected parts of an object file

SYNOPSIS

coffdump [*options*] *file* ...

DESCRIPTION

coffdump dumps selected parts of each of its object file arguments.

coffdump accepts both object files and archives of object files. It processes each file argument according to one or more of the following options.

OPTIONS

- a** Dump the archive header of each member of each archive file argument.
- g** Dump the global symbols in the symbol table of an archive.
- f** Dump each file header.
- o** Dump each optional header.
- h** Dump section headers.
- s** Dump section contents.
- r** Dump relocation information.
- l** Dump line number information.
- t** Dump symbol table entries.
- z *name*** Dump line number entries for the named function.
- c** Dump the string table.
- L** Interpret and print the contents of the *.lib* sections.
- As** Interpret and print the *.sri* section records.
- AS** Interpret and print section contents.
- Am** Interpret and print the *.mir* section records.
- Ai** Interpret and print the contents of the *.inlib* section.
- Ar** Interpret and print the *.rwdi* section records.
- AR** Interpret and print the contents of the *.rwdi* section.
- Aa** Dump the longname table and module table of an archive.

MODIFIERS

Use the following modifiers in conjunction with the options listed above to modify their capabilities.

- d *number*** Dump the section number, *number*, or the range of sections starting at *number* and ending at the *number* specified by **+d**.
- +d *number*** Dump sections in the range either beginning with the first section or beginning with the section specified by **-d**.

- n *name*** Dump information pertaining only to the named entity. This modifier applies to **-h**, **-s**, **-r**, **-l**, and **-t**.
- p** Suppress printing of the headers.
- t *index*** Dump only the indexed symbol table entry. Use **-t** in conjunction with **+t**, to specify a range of symbol table entries.
- +t *index*** Dump the symbol table entries in the range ending with the indexed entry. The range begins at the first symbol table entry or at the entry specified by the **-t** option.
- u** Underline the filename for emphasis.
- v** Dump information in symbolic representation rather than numeric (for example, **C_STATIC** rather than **0X02**). You can use this modifier with all the above options except **-s** and **-o** options of **coffdump**.
- z *name,number***
 Dump line-number entry or range of line numbers starting at *number* for the named function.
- +z *number*** Dump line numbers starting at either function *name* or *number* specified by **-z**, up to *number* specified by **+z**.

Blanks separating an option and its modifier are optional. You can replace the comma separating the name from the number modifying the **-z** option, by a blank.

The **coffdump** command attempts to format the information it dumps in a meaningful way, printing certain information in character, hex, octal, or decimal representation as appropriate.

SEE ALSO

systype(1), **ts(1)**, **a.out(5)**, **ar(5)**

NAME

col – filter reverse line feeds

SYNOPSIS

col [**-bfn**]

DESCRIPTION

col reads the standard input and writes the standard output. It performs the line overlays implied by reverse line feeds (ESC-7 in ASCII) and by forward and reverse half-line feeds (ESC-9 and ESC-8). **col** is particularly useful for filtering multicolumn output made with the **.rt** command of **nroff(1)** and output resulting from using the **tbl(1)** preprocessor.

Although **col** accepts half-line motions in its input, it normally does not emit them on output. Instead, it moves text that would appear between lines to the next lower full-line boundary.

The control characters SO (ASCII code 017), and SI (016) are assumed to start and end text in an alternate character set. **col** remembers the character set (primary or alternate) associated with each printing character read. On output, **col** generates SO and SI characters where necessary to maintain the correct treatment of each character.

All control characters are removed from the input except space, backspace, tab, return, newline, and ESC (033) followed by one of 7, 8, 9, SI, SO, and VT (013). This last character is an alternate form of full-reverse-line feed, for compatibility with some other hardware conventions. **col** ignores all other non-printing characters.

OPTIONS

- b** Assume that the output device in use cannot backspace. If several characters are to appear in the same place, only the last one read is taken.
- f** (fine) Allow the output to contain half-line feeds (ESC-9). Even with this option it never contains either kind of reverse line motion.
- h** Convert white space to tabs to shorten printing time.

BUGS

col can't back up more than 128 lines.
There can be no more than 800 characters, including backspaces, on a line.

SEE ALSO

troff(1), **tbl(1)**

NAME

colcrt – filter **nroff** output for CRT previewing

SYNOPSIS

colcrt [-] [-2] [*file* ...]

DESCRIPTION

colcrt provides virtual half-line and reverse-line feed sequences for terminals without such capability, and for which overstriking is destructive. Half-line characters and underlines (changed to dashes ‘-’) are placed on new lines between the normal output lines.

Normally, **colcrt** uses a minimal space output format that suppresses empty lines. However, the program never suppresses two consecutive empty lines.

OPTIONS

- Suppress all underlining. This option is especially useful for previewing tables that use **tbl**’s **allbox** option.
- 2 Print all half-lines, effectively double-spacing the output. The -2 option is useful for sending output that contains superscripts and subscripts to the line printer (these characters are otherwise invisible).

EXAMPLE

The following example shows a typical use of **colcrt**:

```
tbl exum2.n | nroff -ms | colcrt - | more
```

BUGS

colcrt doesn’t fold underlines onto blanks even with the - option; true underline characters don’t show.

colcrt can’t back up more than 102 lines.

General overstriking is lost. As a special case, | overstruck with a dash (-) or underline becomes a plus sign (+).

Lines are trimmed to 132 characters.

No provision is made for processing superscripts and subscripts in documents that are already double-spaced.

SEE ALSO

col(1), **more(1)**, **nroff(1)**, **troff(1)**, **tbl(1)**, **ul(1)**

NAME

colrm – remove columns from a file

SYNOPSIS

colrm [*startcol* [*endcol*]]

DESCRIPTION

colrm removes selected columns from a file. It takes input from standard input, and sends output to standard output.

If called with one parameter, **colrm** removes the columns of each line starting with the column specified by *startcol*. If called with two parameters, **colrm** removes the columns from the first column (*startcol*) to the last column (*endcol*).

Column numbering starts with column 1.

SEE ALSO

expand(1)

NAME

comb - combine SCCS deltas

SYNOPSIS

comb [-o] [-s] [pSID] [-clist] file ...

DESCRIPTION

comb generates a shell procedure that reconstructs the given SCCS *files*. The reconstructed files are usually smaller than the original *files*. You can specify the arguments in any order, but options apply to all named SCCS files. If you name a directory, **comb** behaves as though each file in the directory is specified as a named file, except that it silently ignores non-SCCS and unreadable files.

If you supply a dash (-) in place of a filename, **comb** reads the standard input. It takes each line of the input to be the name of an SCCS file to be processed, silently ignoring non-SCCS and unreadable files. **comb** writes the generated shell procedure on the standard output.

The valid options are described below. Each is explained as though only one named file is to be processed, but the effects of any option apply independently to each named file. If you do not specify an option, **comb** preserves only leaf deltas and the minimal number of ancestors needed to preserve the tree.

OPTIONS

- clist Specify a *list* of deltas to be preserved. See **get(1)** for the syntax of a *list*.
- o Access the reconstructed file at the release of the delta to be created. This activity occurs for each -e generated. By default, the reconstructed file is accessed at the most recent ancestor. Using this option may decrease the size of the reconstructed SCCS file and alter the shape of the delta tree of the original file.
- pSID Specify the SCCS identification string (SID) of the oldest delta to be preserved. Discard all older deltas in the reconstructed file.
- s Generate a shell procedure which, when run, produces a report on each file. This report contains the filename, size (in blocks) after combining, original size (in blocks), and percentage change computed by the following:

$$100 * (\text{original} - \text{combined}) / \text{original}$$

Use this option before combining any SCCS files, to determine how much space the combining process will save.

BUGS

comb may rearrange the shape of the tree of deltas. It may not save any space; in fact, the reconstructed file may actually be larger than the original.

FILES

s.COMB Reconstructed SCCS file
comb????? Temporary

DIAGNOSTICS

Use **help(1)** for explanations.

SEE ALSO

admin(1), **delta(1)**, **get(1)**, **help(1)**, **prs(1)**, **sccs(1)**, **sh(1)**, **sccsfile(4)**

NAME

comm - select or reject lines common to two sorted files

SYNOPSIS

comm [- [**123**]] *file1 file2*

DESCRIPTION

comm reads *file1* and *file2*, which should be ordered in ASCII collating sequence, and produces a three-column output: lines only in *file1*; lines only in *file2*; and lines in both files. If you specify a dash (-) in place of a filename, **comm** reads the standard input.

Flags 1, 2, or 3 suppress printing of the corresponding column. Thus, **comm -12** prints only the lines common to the two files; **comm -23** prints only lines in the first file but not in the second; **comm -123** is a no-op.

SEE ALSO

cmp(1), **diff(1)**, **uniq(1)**

NAME

compact, **uncompact**, **ccat** – compress and uncompress files, and then cat them

SYNOPSIS

```
compact [ name ... ]
uncompact [ name ... ]
ccat [ file ... ]
```

DESCRIPTION

NOTE: The **compact** command is obsolete; it has been replaced by **compress**(1). The **uncompact** command is provided to allow you to restore files compressed by **compact**.

compact compresses the named files using an adaptive Huffman code. If you do not specify a filename, **compact** compacts the standard input to the standard output. **compact** operates as an on-line algorithm. Each time a byte is read, it is encoded immediately according to the current prefix code. This code is an optimal Huffman code for the set of frequencies seen so far. You do not need to prefix a decoding tree to the compressed file, since the encoder and the decoder start in the same state and stay synchronized. Furthermore, **compact** and **uncompact** can operate as filters. In particular,

... | **compact** | **uncompact** |

operates as a very slow no-op.

When you specify an argument *file*, **compact** compacts it, places the resulting file in *file.C*, and removes *file*. The first two bytes of the compacted file code the fact that the file is compacted. This code is used to prohibit recompaction.

The amount of compression to be expected depends on the type of file being compressed. Typical compression values are text (38%), Pascal source (43%), C source (36%), and binary (19%). These values are the percentages of file bytes reduced.

uncompact restores the original file from a file compressed by **compact**. If you do not specify a filename, **uncompact** uncompresses the standard input to the standard output.

ccat concatenates the original file from a file compressed by **compact**, without uncompressing the file.

BUGS

The last segment of the filename must contain fewer than 30 characters to allow space for the .C that is appended.

FILES

*.C Compacted file created by **compact**, removed by **uncompact**

SEE ALSO

compress(1)

NAME

compress, uncompress, zcat – compress and expand data

SYNOPSIS

```
compress [ -f ] [ -c ] [ -v ] [ -b bits ] [ file ... ]
uncompress [ -f ] [ -v ] [ -c ] [ file ... ]
zcat [ file ... ]
```

DESCRIPTION

compress reduces the size of the named files using adaptive Lempel-Ziv coding. Whenever possible, each file is replaced by one with the extension *.Z*, while keeping the same ownership modes, access, and modification times. If you do not specify a file, **compress** compresses the standard input to the standard output. Use **uncompress** or **zcat** to restore compressed files to their original form.

compress uses the modified Lempel-Ziv algorithm popularized in "A Technique for High Performance Data Compression", Terry A. Welch, *IEEE Computer*, vol. 17, no. 6 (June 1984), pp. 8-19. Common substrings in the file are first replaced by 9-bit codes 257 and up. When code 512 is reached, the algorithm switches to 10-bit codes and continues to use more bits until the limit specified by the **-b** flag is reached (default 16). *bits* must be between 9 and 16. You can change the default in the source to allow **compress** to be run on a smaller machine.

After the *bits* limit is attained, **compress** periodically checks the compression ratio. If it is increasing, **compress** continues to use the existing code dictionary. However, if the compression ratio decreases, **compress** discards the table of substrings and rebuilds it from scratch. This allows the algorithm to adapt to the next "block" of the file.

Note that the **-b** flag is omitted for **uncompress**, since the *bits* parameter specified during compression is encoded within the output, along with a magic number to ensure that neither decompression of random data nor recompression of compressed data is attempted.

The amount of compression obtained depends on the size of the input, the number of *bits* per code, and the distribution of common substrings. Typically, text such as source code or English is reduced by 50–60 percent. Compression is generally much better than that achieved by Huffman coding (as used in **pack**), or adaptive Huffman coding (**compact**), and takes less time to compute.

Exit status is normally 0; if the last file is larger after (attempted) compression, the status is 2; if an error occurs, the exit status is 1.

OPTIONS

- f** Forces compression of *file*. This option is useful for compressing an entire directory, even if some of the files don't actually shrink. If you don't specify **-f**, and **compress** is run in the foreground, you are asked whether an existing file should be overwritten
- c** ("cat") Makes **compress/uncompress** write to the standard output; no files are changed. The nondestructive behavior of **zcat** is identical to that of **uncompress -c**.

- v** Causes the printing of the percentage reduction of each file.
- b *bits*** Determines the maximum number of bits to be used in the encoding scheme. *bits* must be between 9 and 16. The default is 16.

DIAGNOSTICS

Usage: **compress** [**-fvc**] [**-b maxbits**] [*file* ...]

Invalid options were specified on the command line.

Missing maxbits

Maxbits must follow **-b**.

file: not in compressed format

The file specified to **uncompress** has not been compressed.

file: compressed with *xx* bits, can handle only *yy* bits

File was compressed by a program that could deal with more *bits* than the compress code on this machine. Recompress the file with smaller *bits*.

file: already has .Z suffix -- no change

The file is assumed to be already compressed. Rename the file and try again.

file: filename too long to tack on .Z

The file cannot be compressed because its name is longer than 12 characters. Rename and try again.

file already exists; do you wish to overwrite (y or n)?

Respond "y" if you want the output file to be replaced; "n" if not.

uncompress: corrupt input

A SIGSEGV violation was detected which usually means that the input file is corrupted.

Compression: *xx.xx*%

Percentage of the input saved by compression. (Relevant only for **-v**.)

-- not a regular file: unchanged

When the input file is not a regular file, (for example, a directory), it is left unchanged.

-- has *xx* other links: unchanged

The input file has links; it is left unchanged. See **ln(1)** for more information.

-- file unchanged

No savings is achieved by compression. The input remains unchanged.

CAUTIONS

Files that were compressed using the older **compact(1)** utility must be expanded with **uncompact(1)**.

Although compressed files are compatible between machines with large memory, you should use **-b12** for file transfer to architectures with a small process data space (64KB or less).

Ideally **compress** would be more flexible about the existence of the .Z suffix.

NAME

cp – copy files

SYNOPSIS

cp [**-CcfiopPsv**] *file1 file2*

cp [**-CcfiopPrsv**] *file ... directory*

DESCRIPTION

cp copies *file1* onto *file2*. By default, **cp** preserves the mode and owner of *file2* if *file2* already exists; otherwise it uses the mode of the source file modified by the current **umask**(2) is used.

In the second form, one or more *files* are copied into the *directory* with their original filenames.

cp refuses to copy a file onto itself.

OPTIONS

- l** Prompt the user with the filename whenever the copy will cause an old file to be overwritten. An answer of 'y' causes **cp** to continue. Any other answer prevents it from overwriting the file.
- p** Attempt to preserve (duplicate) in copies the modification times and modes of the source files, ignoring the present **umask**.
- r** If any of the source files is a directory, copy each subtree rooted at that name; in this case the destination must be a directory.

Domain/OS BSD OPTIONS

- C** Change the names of any existing files which would have been overwritten. The current date is appended to the filename, in the format *@mm.dd[.n]*. If this name already exists, an additional number is appended. If you specify the **-C** option, the files copied in will not adopt the mode and owner of the existing files.
- c** Change the names of any existing files which would have been overwritten. The current date is appended to the filename, in the format *.mm.dd[.n]*. If this name already exists, an additional number is appended. If you specify the **-c** option, the files copied in do not adopt the mode and owner of the existing files.
- f** Force locked files to be overwritten, and mark the overwritten files to be deleted when they become unlocked. If you specify the **-f** option, the files copied in do not adopt the mode and owner of the existing files.
- o** Copy each file as a typed object, without attempting to open a stream to the file. This is useful in cases where opening a stream would succeed, but not yield the entirety of the underlying object.
- s** Treat symbolic links as files to be copied, rather than copying the destination of the link. This is especially useful when the **-r** option is used to copy an entire directory tree that may contain links to other file systems.

- v** Print the name of each file copied, on the standard output (verbose).
- P** Attempt to preserve any extended access control list (ACL) on the source files, ignoring both the present **umask** and the destination ACL.

SEE ALSO

cat(1), cpacl(1), mv(1), rcp(1C), acl(7)

NAME

cpacl – copy access control list

SYNOPSIS

cpacl [**-odfitvLR**] *source destinations* ...

DESCRIPTION

cpacl copies access control lists (ACLs). If you do not specify an option, **cpacl** assumes **-o**.

The first argument specifies the object from which to copy the ACL. This ACL is then applied to the remaining files and directories as specified. Since only directories have fields specifying ACL inheritance, use of the **-d**, **-f**, and **-i** options requires that the source object be a directory. The destinations can be anything, and the appropriate set of fields are applied (see below).

OPTIONS

- o** Copies the object ACL associated with *source* to each of *destinations*. This is the default if you do not specify an option.
- d** Copies the initial directory ACL associated with the specified directory to all *destinations* that are directories.
- f** Copies the initial file ACL associated with the specified directory to all *destinations* that are directories.
- i** Copies an initial default ACL onto an object ACL. The source must be a directory; destinations that are files receive the initial file ACL of the source, and directories receive the initial directory ACL of the source. If you use **-i** in combination with the **-d** or **-f** options, **cpacl** copies the initial directory or initial file ACL in the usual way.
- t** Uses the object ACL of the source as a template for the initial file or directory ACL of the destination. If the source is a file, **cpacl** copies its ACL to the initial file ACL of any directories. If the source is a directory, **cpacl** copies its ACL only to the initial directory ACL of target directories.
- v** Produce verbose output; that is, show the name of each target as an ACL is copied to it.
- L** Directs **cpacl** to follow any soft links encountered, and operate on the object to which the link points. Since soft links in Domain/OS do not have ACLs, attempting to copy to or from a soft link without the **-L** flag produces just a warning, with no change.
- R** Recursively descends any directories encountered among the destination objects. Be very careful when you combine this option with the **-L** option!

EXAMPLES

To copy the ACL on **foo** to objects **bar** and **zap**, enter the following:

```
cpacl foo bar zap
```

To copy the ACL, initial directory ACL, and initial file ACL on the **/usr/u/fred** directory to all objects in the current directory (files will have only the object ACL applied), enter the following:

```
cpacl -odf /usr/u/fred *
```

To copy the initial file ACL on **fred** to all files in that directory, and the initial directory ACL on **fred** to all sub-directories (also, copy the initial file and directory ACLs to the initial file and directory ACLs of any subdirectories), enter the following:

```
cpacl -idf /usr/u/fred /usr/u/fred/*
```

To copy the object ACL on **file-template** to the initial file ACL of the current directory, enter the following:

```
cpacl -t /usr/u/fred/file-template
```

To copy the object ACL on **dir-template** to the initial directory acl of the current directory, enter the following:

```
cpacl -t /usr/u/fred/dir-template
```

SEE ALSO

cp(1), **chacl(1)**, **lsacl(1)**, **dbacl(1)**, **chmod(1)**, **chgrp(1)**, **chorg(1)**, **chown(8)**, **acl(7)**, **umask(2)**, **salacl(8)**

NAME

cpscr – copy the current display to a file

SYNOPSIS

cpscr [**-inv**] [**-append**] [**-gpr[_bitmap]**] *pathname*

DESCRIPTION

cpscr copies the current screen image, without clearing it, to the file you specify. Use the **prf** (**print_file**) command to print the file.

Use the DM command **cpo** to copy the screen without creating a new process window or changing the current transcript pad. **cpo** invokes the **cpscr** command from the DM without creating a pad or window. Thus, press <CMD> and type

cpo /usr/apollo/bin/cpscr *pathname*

You may copy small portions of a black and white screen (such as a single window) with the DM command **xl**. By default, black and white screens are copied into a GMF file. Color screens are copied into a GPR bitmap.

pathname (required) Specify file to that the screen is copied.

OPTIONS

- inv** Invert image. Use this option to store the image in reverse video. Black screen pixels become white and white screen pixels become black. Do not used this option with the **-gpr_bitmap** option or on color nodes.
- append** Appends a black and white screen image to an existing GMF file. You cannot use this option with the **-gpr_bitmap** option or on color nodes.
- gpr_bitmap** Use this option to copy a black and white screen into a GPR bitmap file rather than a GMF file. This option has no meaning for color nodes since color screens are already copied into GPR bitmaps.

EXAMPLES

Invert and copy the current screen image to the specified file. Since the command line is echoed in the shell's process transcript pad prior to execution, this command will appear in the resulting image.

```
$ cpscr -inv //us/looky_there
<cmd>
```

Command: **cpo /usr/apollo/bin/cpscr -inv //us/looky_there**

Same result as in the previos example, but the **cpscr** line will not appear in the plotted output.

NAME

crddf – create, display, or modify a device descriptor file

SYNOPSIS

crddf [*options ...*] *pathname*

DESCRIPTION

crddf creates, displays, or modifies a device descriptor file (DDF). A DDF defines a peripheral bus unit (PBU) or Small Computer System Interface (SCSI) device for which you have written a driver. See *Writing Device Drivers with GPIO Calls* for details on both DDFs and PBUs.

pathname (required) Specifies name of the DDF to be created, modified, or displayed.

OPTIONS

- Reads further options from standard input. Signal completion with **–end**.
- at** Specifies that the device lives on the AT-compatible bus. Not valid with the **–scsi** option.
- bus *bus_number*** (required for SCSI) Specifies the SCSI bus number. Must be specified with the **–scsi** option. Bus numbering starts at 0; therefore, *bus_number* should always be 0 on a device with one SCSI bus.
- call_library *pathname*** (required for PBU) Specifies *pathname* of the call side of the device driver library. Not valid with the **–scsi** option.
- check** Checks the DDF to ensure that all required fields have been specified.
- cleanup_routine [*entry_name*]** Specifies the entry-point name of the clean-up routine in the call library. Omitting the entry name deletes a previously existing clean-up routine. Not valid with the **–scsi** option.
- csr_offset *port_number*** Specifies the offset into the control status register (CSR) page, in hexadecimal format, at which the device's control/status registers are located. PBU Device drivers may use this information during controller initialization. Not valid with the **–scsi** option.
- csr_page *iova vme_address modifier*** (Required for PBU) Specifies the hexadecimal address of the CSR page for the device in the bus address space. Not valid with the **–scsi** option. The following information applies to the particular bus structure implemented on your system:

- Multibus: optional
- VME bus: optional. If specified, must be page-aligned and in the range C000-D000. DDFs for devices on the DN10000 may also include an optional VME address modifier. If included, it must be set to either A16, A24, or A32. This argument specifies the VME address modifier placed on the bus during communication with the device through the CSR pages only. Specifying the VME address modifier results in a Version 5 ddf being created.
- AT-compatible bus: If specified, may indicate a range (for example, `-csr_page 200 21F`). If the second parameter is missing, a range of 8 consecutive bytes is assumed (for example, `-csr_page 200` assumes a range of 200-207).

<code>-debug</code>	Sets a flag that can be used to turn on debugging logic in a driver. Not valid with the <code>-scsi</code> option.
<code>-display</code>	Displays the current contents of the DDF.
<code>-dma_channel channel-number</code>	Specifies to the driver the DMA channel number used by AT-compatible device. Not valid with the <code>-scsi</code> option. This is a Version 3 option.
<code>-end</code>	Closes the updated DDF and exits.
<code>-initialization_routine entry_name</code> (required for PBU)	Specifies the entry-point name of the initialization routine in the call library. Not valid with the <code>-scsi</code> option.
<code>-interrupt_library pathname</code>	Specifies the pathname of the interrupt side of the device driver library. Not valid with the <code>-scsi</code> option.
<code>-interrupt_routine level [entry_name]</code> (required for PBU)	Specifies a level at which the device interrupts and the entry-point name of an optional interrupt routine. Not valid with the <code>-scsi</code> option.
<code>-major device_number</code>	Specifies the DDF's major device number in range 0-31. Cannot be specified with the <code>-type</code> option.
<code>-minor device_number</code>	Specifies the DDF's minor device number in range 0-511.

- memory_base *iova*** Specifies the MULTIBUS address that marks the base of a controller's local memory. If the specified *iova* is less than 64K this is a Version 2 option, if *iova* is greater than 64K, this is a Version 3 option. Not valid with the **-scsi** option.
- memory_size *length*** Specifies the size, in hexadecimal format, of the controller memory. If the specified *iova* less than 64K, this is a Version 2 option; if greater than 64K, this is a Version 3 option. Not valid with the **-scsi** option.
- multiple** Specifies that the device driver supports more than one device and cause the **crddf** command to check the driver entry-point names listed in the DDF for each device to ensure that it doesn't load multiple copies of the same driver. Not valid with the **-scsi** option.
- node[f] {*node number*}* }** (required for PBU) Specifies the hexadecimal node ID of the node to which the device is physically connected. **-nodef** suppresses the check which makes certain the node exists. You may use an asterisk (*) instead of the node ID to indicate the local node. Not valid with the **-scsi** option.
- quit** Exits without modifying the original DDF.
- replace** Replaces (i.e., overwrite) an existing DDF with a new version. To modify only selected portions of an existing DDF, use **-update**.
- revision [*string*]** Specifies an optional revision number as an 8-character string.
- scsi** Specifies the DDF is for a SCSI device driver.
- serial_number [*string*]** Specifies an optional serial number as a 16-character string.
- share** Specifies a DDF for a controller that can be shared among multiple processes.
- stack_size [*decimal number*]** Specifies the number of bytes, in decimal, to be allocated to the interrupt stack (default is 1024).
- target *target_id*** (required for SCSI) Specifies the *target_id* of the SCSI device. Must be specified with the **-scsi** option.
- type *type name*** Specifies the DDF's type. The type must already be installed on the node. Cannot be specified with the **-major** option.
- unit *unit number*** (required for PBU) Specifies the unit number of the device (must be equal

	to the lowest interrupt level on which the device interrupts). Not valid with the <code>-scsi</code> option.
	• MULTIBUS: Must be in range 0-5. • VME bus: Must be in range 8-14. • AT-compatible bus: Must be in range 0-15.
<code>-update</code>	Modifies selected portions of an existing DDF. If this option is specified, it must precede all other options on the command line. To replace a DDF completely, use <code>-replace</code> .
<code>-user_info [string]</code>	Specifies up to 64 characters of optional user information (no embedded blanks). Not valid with the <code>-scsi</code> option.
<code>-vme</code>	Specifies that device lives on VME bus. Not valid with the <code>-scsi</code> option. This is a Version 3 option.
<code>-20_bit_addressing</code>	Specifies 20-bit memory address size of controller. Not valid with the <code>-scsi</code> option. The device driver must use PBU2 calls if its DDF specifies this option.

EXAMPLES

1. Create a new DDF specifying only the required information.

```
$ crddf /dev/mt0 -
New DDF.
> -unit 3
> -node 2F
> -csr_page 1400
> -call_library /lib/mt.lib
> -initialization_routine mt_$init
> -interrupt_library /lib/mt.int.lib
> -interrupt_routine 3 mt_$int
> -check
No missing fields.
> -end
$
```

2. Display a DDF.

```
$ crddf /dev/mt0 -display
ddf version: 1
device uid: 00030003 0000002F (unit 3, node 2F)
csr page iova: 1400
call library: /lib/mt.lib
interrupt library: /lib/mt.int.lib
initialization entry point: mt_$init
```

```

cleanup entry point:      mt_$cleanup
interrupt stack size: 1024
interrupt routines:
  level 0: [unused]
  level 1: [unused]
  level 2: [unused]
  level 3: mt_$int
  level 4: [unused]
  level 5: [unused]
  level 6: [unused]
  level 7: [unused]
  serial number:
  revision:
  user info:
$

```

3. Change the name of the interrupt routine in an existing DDF.

```
$ crddf /dev/mt0 -update -interrupt_routine 3 mt_$sio
```

4. Create a new DDF for a device that will be accessed through streams for the installed type foodev:

```

$ crddf /dev/foodev -
New DDF.
> -unit 3
> -node *
> -csr_page 1400
> -call_library /lib/foodev.lib
> -initialization_routine foodev_$init
> -interrupt_library /lib/foodev.int.lib
> -interrupt_routine 3 mt_$int
> -type foodev
> -check
No missing fields.
> -end
$

```

5. Create a DDF for a SCSI device driver:

```

$ crddf foo -scsi -
New DDF.
> -bus 0
> -target 0
> -end
$

```

NAME

crp – create a process on a remote node

SYNOPSIS

crp *-on node_spec* [*options*] [*command line*]

DESCRIPTION

crp creates a process on a remote node.

-on node_spec (required) Specifies the node on which the process is to be created.

command line (optional) Specify command line to be executed by the remote process. If the command string contains embedded blanks, enclose it in quotation marks.

OPTIONS

-cp (default) Create a remote process running with standard streams connected to the current server. The option is not valid if **-cpo** or **-cps** is specified.

-nwp Do not create a window-pane legend indicating that the local window is connected to a remote process. Use with the **-cp** option only.

-cpo Create a remote process without a connection to the current window, and an identity of **user.none.none**. This option is not valid if **-cp** or **-cps** is specified.

-cps Create a remote process without a connection to the current window, and an identity of **user.server.none**. This option is not valid if **-cp** or **-cpo** is specified.

-n name Specify the name of the remote process. This allows remote processes to be traced to their originator. If this option is not specified, the default is **user_id.node_id**.

-login name [*password*] Specify the log-in sequence for the remote process on the command line. If the password is omitted, the system prompts you for it. A null (zero-length) password is specified by the null string.

You may use **-login** with either **-cp**, **-cpo**, or **-cps**. If **-login** is specified with either **-cpo** or **-cps**, the identity of the created process is the same as that of the caller (as opposed to **user.none.none** or **user.server.none**, respectively). This means that **-cpo** and **-cps** are identical if **-login** is also specified.

If you use **-login** with **-cpo** or **-cps**, you must place both the name and the password on the command line. No prompting is available in this case.

- me** Specified instead of **-login**. If **-me** is specified, the created process on the remote node inherits the caller's working directory, naming directory, home directory, and SID. If **-me** is specified with either **-cpo** or **-cps**, the identity of the created process is also that of the caller's (as opposed to **user.none.none** or **user.server.none**, respectively). This means that **-cpo** and **-cps** are identical if **-me** is also specified.
- quiet** Suppress connection/disconnection messages.

EXAMPLES

Create a process on node **c7e8** running the shell, and login with the user ID **Joe**.

```
$ crp -on c7e8 -login Joe
```

Create a process on node **fd5e** running the shell, and inherit the current environment.

```
$ crp -on fd5e -me
```


NAME

crpad – create a transcript pad and window

SYNOPSIS

crpad [*options*] [*pathname*]

DESCRIPTION

crpad creates a transcript pad, copies a file (or standard input) into that pad, and then opens a window into the pad. This new pad is not related to the transcript pad attached to processes running the shell; it is for viewing file contents only. This is primarily useful for displaying output being produced inside a pipeline without interrupting the flow of control in the pipe.

You cannot edit transcript pads. If you wish to place a file in a pad for editing, use the EDIT key or the DM command **ce**.

crpad -input behaves differently. This creates an edit pad and lets you create whatever text you want. When you close the edit pad (with **wc** or the EXIT key), that text is copied to standard output.

pathname (optional) Specify the file to be copied into the pad. Not valid if **-input** is used.

Default if omitted: copy standard input

OPTIONS

-i[nput] Copies data from a temporary edit window to standard output. Not valid if **-tee** or **-pn** are specified.

-p[n] *pathname* Specify a pathname for the pad. If you specify a pathname, the pad is saved in that file. Note that you can also save the pad after it is created by using the DM command **pn** (*pad_name*).

-t[ee] Copy output to standard output in addition to the new pad.

EXAMPLES

Create a pad that displays the file **test.data**.

```
$ crpad test.data
```

Display the intermediate results in a pipeline.

```
$ $grep 256- phone.book | crpad -tee | sort >phone.book.local
```

Create an edit pad. When the pad is closed, sort the text edited and display it in a transcript pad.

```
$ crpad -input | sort | crpad
```

NAME

crty – create a new type

SYNOPSIS

crty [*options*] *type_name*

DESCRIPTION

crty creates a new type. It creates an identifier for the new type, and associates it with the supplied type name. New types are used to identify a new kind of manager for streams.

Named traits are a new feature at SR10.1. In order to use a named trait, you may need to reformat your system's types file.

type_name (required)

Specify the name to assign to the created type.

OPTIONS

-n *node_spec*

Specify the node on which the type is to be created. You may also specify the entry directory of a volume mounted for software installation, as shown in the example below. If this option is omitted, the type is created on the current node.

-l

List the type name/type identifier pair that is created.

-b[*in*ary] *pathname*

Create the type from the specified object module (which was created by **crt**yobj). This allows you to use an object module (shipped on media like floppies, magnetic tapes, etc) to add a new type to a system.

-u *high.low*

Create the type with the specified unique identifier (UID). Give the high and low addresses for the UID as indicated.

Note: Use this option only for system debugging. Misuse of this option may cause programs to behave incorrectly.

-t[*rait*]

Create a named trait instead of a type.

-f[*format*]

Systems before SR10.1 did not support named traits. They need to have the types file reformatted when you first create a named trait. Use this option to reformat the types file. The "-format" option has no effect if the types file already has the new format.

Reformatted types files are upwards compatible with older types files, and should not cause interoperability problems between nodes. Once reformatted, however, they can not be converted back to the older format. Older nodes will see named traits as types.

-r[eplace]

Use this option when the type or trait name already exists on your system, and you wish to change its identifier.

Use this option carefully. If a type or trait has the wrong identifier associated with its name, you may lose the ability to access files or other system features. You may also lose the ability to share information with other nodes.

EXAMPLES

```
$ crty example_type -l
"example_type" 24BF9F41.100001FB created.
```

```
$ crty example_type -n //test_vol -l
"example_type" 24BFA6F8.200001FB created on volume //test_vol.
```

In the following example, the disk has been mounted for software installation. The disk's top level directory (cataloged as `/mount_disk` by the `mount(8)` command) must contain a `sys` directory. If it does not, you get a "type manager directory not found" error.

```
$ /etc/mount /mount_disk
$ crty example_type -n /mount_disk -l
"example_type" 24BFB71E.200001FB
  created on volume //my_node/mount_disk.
```

SEE ALSO

`dlty(1)`, `inty(1)`, `lty(1)`, `trty(1)`, `mount(8)`

NAME

crtyobj – create a type object module for binding

SYNOPSIS

crtyobj [*options*] *type_name* [*variable_name*]

DESCRIPTION

crtyobj creates an object module that contains a global symbol with the type UID. This module is bound with type managers. The variable is passed into calls to **trait_\$mgr_dcl** to declare support for the specified type.

type_name (required) Specify the name of the type for which an object module is to be created.

variable_name (optional) Specify the variable name for the type UID.

Default if omitted: name the variable *type_name_\$uid*

OPTIONS

-b *bin_name* Specify the output binary file name. The default is *type_name.bin*.

-sect *section_name* Specify the section name for the data area in which the variable is declared. The default section name is **.data**.

-u *high.low* Specify the type UID explicitly with the high and low addresses in the positions indicated.

NOTE: Use this option only for system debugging.

EXAMPLES

```
$ crtyobj example_type example_$uid
```

```
$ bind -b example_mgr example_main.bin example_calls.bin example_type.bin
```

SEE ALSO

crty(1), dlty(1), lty(1)

NAME

csh – a shell (command interpreter) with C-like syntax

SYNOPSIS

csh [**-cefinstvVxX**] [**-Dname=value**] [*arg* ...]

DESCRIPTION

csh is a command-language interpreter that incorporates a history mechanism (see History Substitutions), job control facilities (see Jobs), interactive filename and username completion (see Filename Completion), and a C-like syntax.

csh begins by executing commands from the **.cshrc** file in your home directory. If this is a log-in shell, then it also executes commands from your **.login** file. CRT users often place an **stty crt** command in their **.login** files, and also invoke **tset(1)** there. To be able to use its job control facilities, users of **csh** must (and automatically) use the tty driver fully described in **tty(4)**. This tty driver allows you to generate interrupt characters from the keyboard to tell jobs to stop. See **stty(1)** for details on setting options in the tty driver.

Normally, the shell then begins reading commands from the terminal, prompting with **"% "**. Upon reading a line of command input, the shell breaks it into *words*, places this sequence of words on the command-history list, parses it, and then executes each command in the line.

When a log-in shell terminates, it executes commands from the **.logout** file in your home directory.

LEXICAL STRUCTURE

Usually, **csh** splits input lines into words at blanks and tabs. The following, however, are exceptions to this:

- The characters: **&**, **|**, **;**, **>**, **<**, **(**, and **)** form separate words. If doubled in **&&**, **||**, **<<**, **>>**, these pairs form single words. You can make these parser metacharacters part of other words or prevent their special meaning by preceding them with a **** (backslash) character. A newline preceded by a backslash is equivalent to a blank.
- Strings enclosed in matched pairs of quotation marks, **'**, **`**, or **"**, form parts of a word; metacharacters in these strings, including blanks and tabs, do not form separate words. Within pairs of **'** or **"** characters, a newline preceded by a backslash gives a true newline character.
- When the shell's input is not a terminal, the **#** character introduces a comment that continues to the end of the input line. To prevent this special meaning, you can precede the **#** by a **** or place it in quotations, using **`**, **'**, and **"**.

COMMANDS

A simple command is a sequence of words, the first of which specifies the command to be executed. A simple command or a sequence of simple commands separated by `|` characters forms a pipeline. The output of each command in a pipeline is connected to the input of the next. Sequences of pipelines may be separated by `;` characters, and are then executed sequentially. A sequence of pipelines can be placed in the background by adding an `&` character at the end.

Any of the above may be placed in parentheses to form a simple command (which may be a component of a pipeline, and so on). You can also separate pipelines with `||` or `&&` characters to indicate, as in the C language, that the second is to be executed only if the first fails or succeeds, respectively (see Expressions).

JOBS

`csch` associates a job with each pipeline. It keeps a table of current jobs, printed by the `jobs` command, and assigns them small integer numbers. When a job is started asynchronously with an `&`, `csch` prints a line similar to the following:

```
[1] 1234
```

This indicates that the job, which was started asynchronously, was job number 1 and had one (top-level) process whose process ID was 1234.

To suspend a running job, you must send it a stop signal, usually with `CTRL/Z`. Once `csch` has indicated that the job has been stopped (and has printed a prompt), you can manipulate the state of this job. You can put it in the background with the `bg` command, or run some other commands and then eventually bring the job back into the foreground with the `fg` command. A suspend takes effect immediately, causing `csch` to discard pending output and unread input. There is another special key, `CTRL/Y`, which does not generate a `STOP` signal until a program attempts to `read(2)` it. Type `CTRL/Y` ahead when you have prepared some commands for a job that you wish to stop after the program has read them. `CTRL/Y` is not supported in the Display Manager.

A job being run in the background stops if it tries to read from the terminal. Background jobs are normally allowed to produce output, but you can disable this by specifying the `stty tostop` command. Specifying `stty tostop` causes background jobs to stop when they try to produce output just as they do when they read input.

There are several ways to refer to jobs in the shell. The `%` character introduces a job name. Job number 1, for example, becomes `%1`. Naming a job brings it to the foreground; thus, `%1` is a synonym for `fg %1`, bringing job 1 back into the foreground. Similarly, specifying `%1 &` resumes job 1 in the background. Jobs can also be named by prefixes of the string typed in to start them, if the prefix is unambiguous. For example, `%ex` normally restarts a suspended `ex(1)` job, if there is only one suspended job whose name begins with the string "ex". You can also specify `%?string`, to indicate a job whose text contains *string*, if there is only one such job.

csh maintains a notion of the current and previous jobs. In output pertaining to jobs, it marks the current job with a + and the previous job with a -. The abbreviation %+ refers to the current job, and %- refers to the previous job. For close analogy with the syntax of the **history** mechanism (described below), a %% also represents the current job.

STATUS REPORTING

csh knows immediately when the state of a process changes. It normally informs you whenever a job becomes blocked so that no further progress is possible, but only just before it prints a prompt. (This is so that it does not otherwise disturb your work.) However, if you set the **notify** shell variable, **cs**h immediately reports status changes in background jobs. The **notify** shell command also marks a single process so that its status changes are immediately reported. By default, **notify** marks the current process. Thus, you only have to type **notify** after starting a background job to mark it.

When you try to leave the shell while jobs are stopped, you are warned that "You have stopped jobs." You can use the **jobs** command to see which jobs are stopped. A second attempt to exit causes the suspended jobs to terminate without warning.

FILENAME COMPLETION

When the filename completion feature is enabled by setting the shell variable **filec** (see **set**), **cs**h interactively completes filenames and usernames from unique prefixes, when they are input from the terminal followed by the escape character (the escape key, or CTRL/L). For example, if the current directory looks like

DSC.OLD	bin	cmd	lib	xmpl.c
DSC.NEW	chaosnet	cmtest	mail	xmpl.o
bench	class	dev	mbx	xmpl.out

and the input is

```
% vi ch<ESC>
```

csh completes the prefix "ch" to the only matching filename "chaosnet", changing the input line to

```
% vi chaosnet
```

However, if you specify

```
% vi D<ESC>
```

csh expands the input only to

```
% vi DSC.
```

and sounds the terminal bell to indicate that the expansion is incomplete, because two filenames match the prefix "DSC".

If a partial filename is followed by the end-of-file character (usually CTRL/D), then, instead of completing the name, **cs**h lists all filenames matching the prefix. For example, the input

```
% vi D<CTRL/D>
```

causes all files beginning with "D" to be listed:

```
DSC.NEW DSC.OLD
```

while the input line remains unchanged.

You can use the same system of escape and end-of-file to expand partial usernames, if the word to be completed (or listed) begins with the character "'~". For example, typing

```
cd ~ro<CTRL/D>
```

may produce the expansion

```
cd ~root
```

Set the variable **nobeep**, to inhibit the use of the terminal bell to signal errors or multiple matches.

Normally, all files in the particular directory are candidates for name completion. Files with certain suffixes can be excluded from consideration by setting the variable **ignore** to the list of suffixes to be ignored.

Thus, if you set **ignore** by the command

```
% set ignore = (.o .out)
```

then typing

```
% vi x<ESC>
```

results in the completion to

```
% vi xmpl.c
```

ignoring the files **xmpl.o** and **xmpl.out**. However, if the only completion possible requires not ignoring these suffixes, they are not ignored. Also, **ignore** does not affect the listing of filenames by CTRL/D. All files are listed regardless of their suffixes.

HISTORY SUBSTITUTIONS

History substitutions place words from previous command input as portions of new commands, making it easy to repeat commands, repeat arguments of a previous command in the current command, or fix spelling mistakes in a previous command with little typing and much confidence. History substitutions begin with the character **!** and can start anywhere in the input stream (providing that they do not nest). Precede the **!** with a **** to prevent its special meaning. For convenience, a **!** is passed unchanged when it is followed by a blank, tab, newline, **=**, or **(**. History substitutions also occur when an input line begins with a **^** (see below). Before being executed, input lines containing history substitution are echoed on the terminal as they could have been typed without history substitution.

csh saves input commands consisting of one or more words on the history list. The history substitutions reintroduce sequences of words from these saved commands into the input stream. The size of the history list is controlled by the **history** variable. The previous command is always retained, regardless of its value. Commands are numbered sequentially from 1.

For example, consider the following output from the **history** command:

```
9  write michael
10 ex write.c
11 cat oldwrite.c
12 diff *write.c
```


The commands are shown with their event numbers. It is not usually necessary to use event numbers, but you can make the current event number part of the prompt by placing an **!** in the **prompt** string.

Supposing the current event is 13, you can refer to previous events by event number, as in **!11** for event 11; relatively, as in **!-2** for event 11; by a prefix of a command word, as in **!d** for event 12 or **!wrl** for event 9; or by a string contained in a word in the command, as in **!?mic?**, also referring to event 9.

These forms, without further modification, simply reintroduce the words of the specified events, each separated by a single blank. As a special case, **!!** refer to the previous command. Thus, **!!** alone is essentially a redo.

To select words from an event, you can follow the event specification by a **:** and a designator for the desired words. The words of an input line are numbered from 0, the first (usually command) word being 0, the second word (first argument) being 1, and so on. The basic word designators are:

0	first (command) word
<i>n</i>	<i>n</i> 'th argument
^	first argument, i.e. '1'
\$	last argument
%	word matched by (immediately preceding) ?s? search
<i>x</i> - <i>y</i>	range of words
- <i>y</i>	abbreviates '0- <i>y</i> '
*	abbreviates '^-\$', or nothing if only 1 word in event
<i>x</i> *	abbreviates ' <i>x</i> -\$'
<i>x</i> -	like ' <i>x</i> * ' but omitting word '\$'

The **:** separating the event specification from the word designator can be omitted if the argument selector begins with a **^**, **\$**, ***** or **%**. After the optional word designator can be placed a sequence of modifiers, each preceded by a **:**. The following modifiers are defined:

h	Remove a trailing pathname component, leaving the head.
r	Remove a trailing ' .xxx ' component, leaving the root name.
e	Remove all but the extension ' .xxx ' part.
s/l/r/	Substitute <i>l</i> for <i>r</i>
t	Remove all leading pathname components, leaving the tail.
&	Repeat the previous substitution.
g	Apply the change globally, prefixing the above, e.g. ' g& '.
p	Print the new command but do not execute it.
q	Quote the substituted words, preventing further substitutions.
x	Like q , but break into words at blanks, tabs and newlines.

Unless preceded by a **g**, the modification is applied only to the first modifiable word. With substitutions, it is an error for no word to be applicable.

The left-hand side of substitutions are not regular expressions in the sense that they are in the editors (**ed**, **vi**, and so on) rather they are strings. You can use any character as the delimiter in place of a **/**. A **** quotes the delimiter into the *l* and *r* strings. The character **&** on the right-hand side is replaced by the text from the left. A **** also

quotes an `&`. A null `/` uses the previous string either from an `/` or from a contextual scan string `s` in `!s?`. The trailing delimiter in the substitution, as well as the trailing `?` in a contextual scan, can be omitted if a newline follows immediately.

You can specify a history reference without an event specification, for example, `!$`. In this case, the reference is to the previous command unless a previous history reference occurred on the same line (in which case this form repeats the previous reference).

Thus, `!foo?`!$'` gives the first and last arguments from the command matching `?foo?`.

A special abbreviation of a history reference occurs when the first non-blank character of an input line is a `^`. This is equivalent to `!s^` and provides a convenient shorthand for substitutions on the text of the previous line. Thus, `!b^lib` fixes the misspelling of `lib` in the previous command. Finally, a history substitution may be surrounded with `{ and }` if necessary, to insulate it from the characters that follow. Thus, after `ls -ld ~paul` you might type `!{l}a` to do `ls -ld ~paula` while `!la` looks for a command starting with `la`.

QUOTATIONS WITH SINGLE AND DOUBLE QUOTES

Placing strings in single and double quotation marks prevents all or some of the remaining substitutions. Those enclosed in single quotation marks are prevented any further interpretation; those in double quotation marks may be expanded as described below.

In both cases, the resulting text becomes all or part of a single word. In only one special case (see `COMMAND SUBSTITUTION` below) does a double-quoted string yield parts of more than one word; single-quoted strings never do.

ALIAS SUBSTITUTION

`cs`h maintains a list of aliases that can be established, displayed, and modified by the `alias` and `unalias` commands. After it scans a command line, `cs`h parses the line into distinct commands and checks the first word of each command, left-to-right, for an alias. If it finds one, it rereads the text that is the alias for that command (with the history mechanism available) as though that command were the previous input line. The resulting words replace the command and argument list. If no reference is made to the history list, `cs`h leaves the argument list unchanged.

For example, if the alias for `ls` is `ls -l`, the command `ls /usr` maps to `ls -l /usr` and the argument list is undisturbed. Similarly, if the alias for `lookup` is `grep !` /etc/passwd`, then `lookup bill` maps to `grep bill /etc/passwd`.

Every time `cs`h finds an alias, it transforms the input text and begins the aliasing process again on the reformed input line. Prevent looping (if the first word of the new text is the same as the old) by flagging the first word to prevent further aliasing. `cs`h detects other loops and returns an error.

Note that the mechanism allows aliases to introduce parser metasyntax. Thus, you can specify `alias print `pr \!* | lpr`` to make a command that `pr`'s its arguments to the line printer.

VARIABLE SUBSTITUTION

cs**h** maintains a set of variables, each having as value a list of zero or more words. **cs****h** sets some of these variables, and merely refers to others. For instance, the *argv* variable is an image of the shell's argument list, and words of its value are referred to in special ways.

You can display and change the values of variables by using the **set** and **unset** commands. Some of the variables the shell refers to are toggles. The shell does not care what their value is, only whether they are set. For instance, the **verbose** variable is a toggle that causes command input to be echoed. Use the command line option **-v** to set this variable.

Other operations treat variables numerically. The command represented by the **at** sign, **@**, permits numeric calculations to be performed, with the result assigned to a variable. However, variable values are always represented as (zero or more) strings. In numeric operations, the null string is considered to be zero, and the second and subsequent words of multiword values are ignored.

After **cs****h** has aliased and parsed the input line, and before executing each command, it performs variable substitution keyed by **\$** characters. You can prevent this expansion by preceding the **\$** with a ****, except within double quotation marks (**"**), where it always occurs, and within single quotation marks (**'**) where it never occurs. Strings enclosed in single quotation marks are interpreted later (see **COMMAND SUBSTITUTION** below), so the dollar sign (**\$**) substitution does not occur until later, if at all. A **\$** is passed unchanged if followed by a blank, tab, or end-of-line.

Input/output redirections are recognized before variable expansion, and are variable-expanded separately. Otherwise, the command name and entire argument list are expanded together. Therefore, the first (command) word to this point can generate more than one word; the first word becomes the command name, and the rest become arguments.

Unless you enclose the results of variable substitution in double quotation marks or specify the **:q** modifier, they may eventually be command and filename substituted. Within double quotation marks, a variable whose value consists of multiple words expands to a portion of a single word, with the words of the variable's value separated by blanks. When the **:q** modifier is applied to a substitution, the variable expands to multiple words with each word separated by a blank and enclosed in quotation marks to prevent later command or filename substitution.

The following metasequences are provided to introduce variable values into the shell input. Except as noted, you cannot reference a variable that is not set. You can apply the **:h**, **:t**, **:r**, **:gh**, **:gt**, and **:gr** modifiers to most of the substitutions below. Substitutions that you cannot do this with are marked accordingly. If braces appear in the command form, you must put the modifiers within the braces. You can apply only one modifier beginning with a colon (**:**) on each expansion preceded by a dollar sign (**\$**).

\$name

\${name}

Replace text by the words of the value of variable *name*, each separated by a blank. Braces insulate *name*

from following characters, which would otherwise be part of it. Shell variables have names consisting of up to 20 letters and digits starting with a letter. The underscore character (`_`) is considered a letter. If *name* is not a shell variable, but is set in the environment, that value is returned. However, colon (`:`) modifiers and the other forms given below are not available in this case.

`$name[selector]`
`${name[selector]}`

Select only some of the words from the value of *name*. The selector is subjected to `$` substitution and may consist of a single number or two numbers separated by a dash (`-`). The first word of a variable's value is numbered 1. If you omit the first number of a range, the number defaults to 1. If you omit the last member of a range, the number defaults to `$#name` . The selector `*` selects all words. It is not an error for a range to be empty if you omit the second argument or it is in range.

`$#name`
`${#name}`

Give the number of words in the variable. This is useful for later use in a `"[selector]"`.

`$0`

Substitute the filename from which command input is being read. An error occurs if the name is not known.

`$number`
`${number}`

This sequence is equivalent to `$argv[number]`.

`$*`

This sequence is equivalent to `$argv[*]`.

`$?name`
 `${?name}`

Substitute the string 1 if *name* is set; 0 if it is not. This substitution cannot be modified with modifiers preceded by a `:`.

`$?0`

Substitute 1 if the current input filename is known; 0 if it is not. This substitution cannot be modified with modifiers preceded by a `:`.

`$$`

Substitute the decimal process number of the parent shell. This substitution cannot be modified with modifiers preceded by a `:`.

`$<`

Substitute a line from the standard input, with no further interpretation. This sequence is useful for reading from the keyboard in a shell script. This substitution cannot be modified with modifiers preceded by a `:`.

`csh` applies the remaining substitutions, command and filename substitution, selectively to the arguments of built-in commands. This means that portions of expressions not evaluated are not subjected to these expansions. Names for commands that are not internal to the shell are substituted separately from the argument list. This

occurs very late, after input/output redirection is performed, and in a child of the main shell.

COMMAND SUBSTITUTION

Enclosing a command in closing quotation marks (q) indicates command substitution. **cs**h usually breaks the output from such a command into separate words at blanks, tabs, and newlines. It discards null words, and uses the modified text to replace the original string. Within double quotation marks, only newlines force new words; blanks and tabs are preserved.

In any case, the single final newline does not force a new word. Note that it is thus possible for a command substitution to yield only part of a word, even if the command outputs a complete line.

FILENAME SUBSTITUTION

If a word contains any of the characters *, ?, [, {, or it begins with ~, that word is a candidate for filename substitution, also known as "globbing." **cs**h regards the word as a pattern, replacing it with an alphabetically sorted list of filenames that match the pattern. In a list of words specifying filename substitution, at least one pattern must match an existing filename, but each pattern need not match. Only the metacharacters *, ?, and [imply pattern matching. The characters ~ and { are like abbreviations.

In matching filenames, you must match a . at the beginning of a filename or immediately following a / explicitly. This is also true for the / itself. An * matches any string of characters, including the null string. A ? matches any single character. The sequence [...] matches any one of the characters enclosed. Within such a sequence, a pair of characters separated by a - matches any character lexically between the two.

The character ~ at the beginning of a filename refers to home directories. Standing alone, it expands to your home directory (reflected in the value of the variable **home**). When the ~ is followed by a name consisting of letters, digits, and -, **cs**h searches for a user with that name and substitutes his home directory. Thus, ~**ken** might expand to **/usr/ken** and ~**ken/chmach** to **/usr/ken/chmach**. If the ~ is followed by a character other than a letter or /, or if it appears somewhere other than at the beginning of a word, the shell leaves it undisturbed.

The metanotation *a{b,c,d}e* is shorthand for *abe ace ade*. Left-to-right order is preserved. The results of matching are sorted separately at a low level to preserve this order (nesting is acceptable). Thus, ~**source/s1/{oldls,ls}.c** expands to **/usr/source/s1/oldls.c /usr/source/s1/ls.c** whether or not these files exist, without any chance of error if the home directory for **source** is **/usr/source**. Similarly, **../{memo,*box}** might expand to **../memo ../box ../mbox**. (Note that **memo** was not sorted with the results of matching ***box**.) As a special case, the shell passes all single unmatched braces or an empty pair of braces undisturbed.

INPUT/OUTPUT

To redirect the standard input and standard output of a command, use the following syntax:

< *name* Open the file *name* (which is first variable-, command-, and filename-expanded) as the standard input.

<< *word*

Read the shell input up to a line identical to *word*. *word* is not subjected to variable-, filename-, or command-substitution. Each input line is compared to *word* before any substitutions are done on this input line. Unless a quoting \, ", or ` character appears in *word*, *cs*h performs variable and command substitution on the intervening lines, allowing \ to quote a \$, a \, and `. Commands that are substituted have all blanks and tabs preserved. All new-lines except for the final one are also preserved. The resulting text is placed in an anonymous temporary file, which is given to the command as standard input.

> *name*

>! *name*

>& *name*

>&! *name*

Use the file *name* as standard output. If the file does not exist, create it; if the file does exist, truncate it, discarding its previous contents.

If the variable **noclobber** is set, the file must not exist, or it must be a character special file (for example, a terminal or */dev/null*), or an error results. This helps prevent accidental destruction of files. The ! forms suppress this check.

Forms involving & route the diagnostic output, as well as the standard output, into the specified file. *name* is expanded in the same way as input filenames beginning with < are.

>> *name*

>>& *name*

>>! *name*

>>&! *name*

Use the file *name* as standard output, like >, but place output at the end of the file. If the variable **noclobber** is set, it is an error for the file not to exist unless you specify one of the forms containing !.

A command receives the environment in which the shell was invoked, as modified by the input/output parameters and the presence of the command in a pipeline. Thus, unlike some previous shells, commands run from a file of shell commands have no access to the text of the commands by default; rather, they receive the original standard input of the shell. The << mechanism should be used to present in-line data. This permits shell command scripts to function as components of pipelines and allows the shell to block-read its input. Note that the default standard input for a command run detached is not modified to be the empty file */dev/null*. Rather, the standard input remains as the original standard input of the shell. If this is a terminal and if the process attempts to read from the terminal, the process blocks and you are notified (see JOBS above.)

Diagnostic output may be directed through a pipe with the standard output. Simply use the form `|&` instead of `|` to do this.

EXPRESSIONS

A number of the built-in commands take expressions that have operators similar to those used for the C language, with the same precedence. These expressions appear in the `@`, `exit`, `if`, and `while` commands. The following operators are available:

`|| && | ^ ^ & == != =~ !~ <= >= < > << >> + - * / % ! ~ ( )`

Here the precedence increases to the right, The following characters are, in groups, at the same level:

```
==  !=  =~  !~
<=  >=  <   >
<<  >>
+    -
*    /  %
```

The following operators compare their arguments as strings:

`== != =~ !~`

All others operate on numbers. The operators `=~` and `!~` are like `==` and `!=` except that the right-hand side is a pattern (containing, for example, asterisks, question marks, and instances of [...] characters) against which the left-hand operand is matched. This removes the need to use the `switch` statement in shell scripts when all you need is pattern-matching.

`cs`h considers strings beginning with a zero to be octal numbers. It interprets null or missing arguments as zero. The results of all expressions are strings, which represent decimal numbers. Note that no two components of an expression can appear in the same word. They should be surrounded by spaces, except when adjacent to components of expressions that are syntactically significant to the parser (`&`, `|`, `<`, `>`, `(`, `)`).

Also available in expressions as primitive operands are command executions enclosed in braces (`{` and `}`), and file enquiries of the form `-l name` where `l` is one of the following:

r	Read access
w	Write access
x	Execute access
e	Existence
o	Ownership
z	Zero size
f	Plain file
d	Directory

`cs`h performs command and filename expansion on the specified name, and then checks to see if it has the specified relationship to the real user. If the file does not

exist, or if it is inaccessible, all inquiries return false (0). Command executions succeed, returning true (1), if the command exits with status 0; otherwise, they fail, returning false (0). If you require more detailed status information, execute the command outside an expression and examine the **status** variable.

CONTROL FLOW

csh contains a number of commands to regulate the flow of control in command files (shell scripts) and (in limited but useful ways) from terminal input. These commands all operate by forcing the shell to reread or skip in its input and, due to the implementation, restrict the placement of some of the commands.

The **foreach**, **switch**, and **while** statements, as well as the **if-then-else** form of the **if** statement require that the major keywords appear in a single simple command on an input line, as shown below.

If the shell's input is not seekable, the shell buffers input whenever a loop is being read and performs seeks in this internal buffer to accomplish the rereading implied by the loop. (To the extent that this allows, backward **gotos** succeed on non-seekable inputs.)

BUILT-IN COMMANDS

Built-in commands are executed within the shell. If a built-in command occurs as any component of a pipeline except the last, it is executed in a sub-shell.

alias Print all aliases.

alias *name*

Print the alias for *name*.

alias *name wordlist*

Assign the specified *wordlist* as the alias of *name*. The *wordlist* is command- and filename-substituted. *name* cannot be **alias** or **unalias**.

alloc Show the amount of dynamic memory acquired, broken down into used and free memory. With an argument, this command shows the number of free and used blocks in each size category. The categories start at size eight and double at each step. This command's output may vary across system types.

bg

bg *%job ...*

Put the current or specified jobs into the background, continuing them if they were stopped.

break Resume execution after the end of the nearest enclosing **foreach** or **while**. Execute the remaining commands on the current line. Thus you can have multi-level breaks by writing them all on one line.

breaksw

Break from a **switch**, resuming after the **endsw**.

case *label:*

Specify a label in a **switch** statement.

cd

cd *name*

chdir

chdir *name*

Change the shell's working directory to directory *name*. If you do not specify an argument, change to the home directory of the user.

If *name* is not found as a subdirectory of the current directory and does not begin with */*, *./*, or *../*, check each component of the variable **cdpath** to see if it has a subdirectory *name*. Finally, if all else fails but *name* is a shell variable whose value begins with */*, check to see if it is a directory.

continue

Continue execution of the nearest enclosing **while** or **foreach**. Execute remaining commands on the current line.

default:

Label the default case in a **switch** statement. This command should follow all **case** labels.

dirs

Print the directory stack. The top of the stack is at the left, and the first directory in the stack is the current directory.

echo *wordlist*

echo **-n** *wordlist*

Write the specified words to the shell's standard output, separated by spaces, and terminated with a newline, unless you specify the **-n** option.

else

end

endif

endsw See the description of the **foreach**, **if**, **switch**, and **while** statements below.

eval *arg ...*

Read the arguments as input to the shell, executing the resulting command(s) in the context of the current shell. This occurs as in **sh**(1). The command is generally used to execute commands generated as the result of command or variable substitution, since parsing occurs before these substitutions. See **tset**(1) for an example of using **eval**.

exec *command*

Execute the specified *command* in place of the current shell.

exit

Exit with the value of the **status** variable.

exit(*expr*)

Exit with the value of the specified *expr*.

fg

fg *%job ...*

Bring the current or specified jobs into the foreground, continuing them if they were stopped.

foreach *name* (*wordlist*)

...
end Successively set the variable *name* to each member of *wordlist*, and execute the sequence of commands between this command and the matching *end*. (Both **foreach** and **end** must appear alone on separate lines.)

Use the **continue** command to continue the loop prematurely. Use the **break** command to terminate it prematurely. When the shell reads this command from the terminal, it reads the loop once, prompting with "? " before executing any statements in the loop. If you make a mistake typing in a loop at the terminal, you can interrupt it.

glob *wordlist*

Perform the same function as the **echo** command, but do not recognize backslash escapes, and delimit words by null characters in the output. Use this command with programs that use the shell to filename-expand a list of words.

goto *word*

Perform filename- and command-expansion on the specified *word* to yield a string of the form *label*. Cause the shell to rewind input as much as possible and search for a line of the form *label*: (possibly preceded by blanks or tabs). Continue execution after the specified line.

hashstat

Print a statistics line indicating how effective the internal hash table has been at locating commands and avoiding instances of the **exec** command. An **exec** is attempted for each component of the path where the hash function indicates a possible hit, and in each component that does not begin with a slash.

history Display the history event list.

history *n*

Print only the *n* most recent events in the history event list.

history -r *n*

Print the most recent events first (rather than printing the oldest first).

history -h *n*

Print the history event list without leading numbers. Use this command to produce files suitable for sourcing with the -h option to the **source** built-in command.

if (*expr*) *command*

If the specified expression evaluates true, execute the single *command* with arguments. Variable substitution on *command* happens early, at the same time it does for the rest of the **if** command. The *command* must be a simple command, not a pipeline, a command list, or a parenthesized command list. Input/output redirection occurs even if *expr* is false, in which case, *command* is not executed.

if (*expr*) **then**

...

else if (*expr2*) **then**

...

else

...

endif If the specified *expr* is true, execute the commands to the first **else**; otherwise if *expr2* is true, execute the commands to the second **else**, and so on. Any number of **else-if** pairs are possible; only one **endif** is needed. The **else** part is optional. (The words **else** and **endif** must appear at the beginning of input lines; the **if** must appear alone on its input line or after an **else**).

inlib *lib* Install a user-supplied library specified by *lib* in the shell process. The library is used to resolve external references of programs (and libraries) loaded after its installation. Note that the library is not loaded into the address space unless it is needed to resolve an external reference. The list of **inlib**ed libraries is passed to all children of the current shell. Use **llib**(1) to examine this list.

jobs List the active jobs.

jobs -l List the active jobs, and include process IDs.

kill *%job*

kill -sig *%job ...*

kill *pid*

kill -sig *pid ...*

kill -l Send either the TERM (terminate) signal or the specified signal to the jobs or processes indicated. Provide signals by number or by names (as given in */usr/include/signal.h*, stripped of the SIG prefix). A **kill -l** lists the signal names. There is no default process for this command. If the signal being sent is TERM (terminate) or HUP (hangup), the job or process is sent a CONT (continue) signal as well.

limit

limit *resource*

limit *resource maximum-use*

limit -h

limit -h *resource*

limit -h *resource maximum-use*

Limits the consumption by the current process and each process it creates, to not individually exceed *maximum-use* on the specified *resource*. If you do not specify *maximum-use*, the current limit is printed; if you do not specify *resource*, all limitations are given. If you specify the **-h** flag, the hard limits are used instead of the current limits. The hard limits impose a ceiling on the values of the current limits. Only the super-user can raise the hard limits, but a user can lower or raise the current limits within the legal range.

Resources controllable currently include **cputime** (the maximum number of CPU seconds to be used by each process), **filesize** (the largest single file that you can create), **datasize** (the maximum growth of the data region via

sbrk(2) beyond the end of the program text), **stacksize** (the size of the children's stack regions), and **coredumpsiz**e (the size of the largest core dump that will be created). NOTE: The **coredumpsiz**e limit is always 0 in Domain/OS.

You can specify the *maximum-use* as a (floating point or integer) number followed by a scale factor. For all limits other than *cpulim*e the default scale is 'k' or 'kilobytes' (1024 bytes); you can also use a scale factor of 'm' or 'megabytes'. For *cpulim*e the default scaling is 'seconds', but you can specify 'm' for minutes or 'h' for hours, or a time of the form 'mm:ss' giving minutes and seconds.

For both *resource* names and scale factors, unambiguous prefixes of the names suffice.

login Terminate a log-in shell, replacing it with an instance of **/bin/login**. This is one way to log out, and it is included for compatibility with **sh(1)**.

logout Terminate a log-in shell. This command is especially useful if **ignoreeof** is set.

nice Set the **nice(1)** scheduling priority for this shell to 4.

nice +number

Set the **nice(1)** priority to the given *number*.

nice command

Run *command* at **nice(1)** priority 4.

nice +number command

Run *command* at positive *number nice(1)* priority. The greater the number, the lower CPU priority the process gets. The super-user can specify negative priority by using **nice -number** The command is always executed in a sub-shell, and the restrictions placed on commands in simple **if** statements apply.

nohup When you specify this command in a shell script, ignore hangups for the remainder of the script.

nohup command

Run the specified *command* with hangups ignored. This happens to all processes detached with **&**.

notify

notify %job ...

Notify the user asynchronously when the status of the current or specified jobs changes (normally, notification is presented before a prompt). This is automatic if the shell variable **notify** is set.

onintr Restore the default action of the shell on interrupts (to terminate shell scripts or to return to the terminal command input level). In any case, if the shell is running detached and interrupts are being ignored, all forms of **onintr** have no meaning, and interrupts continue to be ignored by the shell and all invoked commands.

- onintr -**
Ignore all interrupts.
- onintr *label***
Execute a **goto** *label* when an interrupt is received or a child process terminates because it was interrupted.
- popd** Pop the directory stack, returning to the new top directory. The elements of the directory stack are numbered from zero, starting at the top.
- popd +*n***
Discard the *n*th entry in the directory stack.
- pushd** Exchange the top two elements of a directory stack.
- pushd *name***
Change to *name* directory and push the old current working directory onto the directory stack.
- pushd +*n***
Rotate the *n*th argument of the directory stack around to be the top element and change to it. The members of the directory stack are numbered from zero, starting at the top.
- rehash** Recompute the internal hash table of the contents of the directories in the **path** variable. This is needed if new commands are added to directories in the **path** while you are logged in. This should be necessary only if you add commands to one of your own directories, or if someone changes the contents of one of the system directories.
- repeat *count command***
Execute the specified *command* (subject to the same restrictions as the *command* in the one-line **if** statement above) *count* times. I/O redirections occur exactly once, even if *count* is zero.
- rootnode *arg***
Change the current node entry directory to *arg*. See **rootnode(1)**.
- set** Show the value of all shell variables. Variables that have other than a single word as their value appear as a parenthesized word list.
- set *name***
Set *name* to the null string.
- set *name=word***
Set *name* to the single *word*. In all cases, the value is command- and filename-expanded.
- set *name[index]=word***
Set the *index*th component of *name* to *word*. This component must already exist. In all cases, the value is command- and filename-expanded.
- set *name=(wordlist)***
Set *name* to the list of words in *wordlist*. In all cases, the value is command- and filename-expanded. You can repeat these arguments to set multiple

values in a single **set** command. Note, however, that variable expansion happens for all arguments before any setting occurs.

setenv List all current environment variables.

setenv *name*

Set *name* to an empty string.

setenv *name value*

Set the value of the environment variable *name* to be *value*, a single string. The most commonly used environment variables — **USER**, **TERM**, and **PATH** — are automatically imported to and exported from the **cs**h variables **user**, **term**, and **path**. You do not have to use **setenv** for these.

shift Shift the members of **argv** to the left, discarding **argv[1]**. It is an error for the **argv** variable not to be set or to have less than one word as its value.

shift *variable*

Shift the specified *variable* to the left, discarding *variable[1]*.

source *name*

Read commands from *name*. You may nest **source** commands, but if you nest them too deeply, the shell may run out of file descriptors. An error in a **source** at any level terminates all nested **source** commands.

source **-h** *name*

Place commands in the history list without executing them. Normally, input during **source** commands is not placed on the history list.

stop Stop the current job that is executing in the background.

stop *%job ...*

Stop the specified *job* that is executing in the background.

suspend

Causes the shell to stop in its tracks, much as if it had been sent a stop signal with **CTRL/Z**. This is most often used to stop shells started by **su(1)**

switch (*string*)

case *str1*:

...

breaksw

...

default:

...

breaksw

endsw

Each case label is successively matched, against the specified *string* which is first command and filename expanded. The file metacharacters *****, **?** and **[...]** may be used in the case labels, which are variable expanded. If none of the labels match before a "default" label is found, then the execution begins

after the default label. Each case label and the default label must appear at the beginning of a line. The command **breaksw** causes execution to continue after the **endsw**. Otherwise control may fall through case labels and default labels as in C. If no label matches and there is no default, execution continues after the **endsw**.

time Print a summary of time used by this shell and its children.

time *command*

If arguments are given the specified simple command is timed and a time summary as described under the **time** variable is printed. If necessary, an extra shell is created to print the time statistic when the command completes.

umask Display the file-creation mask (in octal).

umask *value*

Set the file-creation mask to the specified *value*. Common values for the mask are 002 (giving all access to the group and read and execute access to others) or 022 (giving all except write access to users in the group or others).

unalias *pattern*

Discard all aliases whose names match the specified *pattern*. Thus, **unalias *** removes all aliases. It is not an error for nothing to be **unaliased**.

unhash Disable the internal hash table mechanism, normally used to speed location of executed programs.

unlimit

unlimit *resource*

unlimit **-h**

unlimit **-h** *resource*

Removes the limitation on *resource*. If you do not specify *resource*, all *resource* limitations are removed. If you specify **-h**, the corresponding hard limits are removed. Only the super-user can do this.

unset *pattern*

Remove all variables whose names match the specified pattern. Thus, **unset *** removes all variables. This has noticeably distasteful side-effects. It is not an error for nothing to be **unset**.

unsetenv *pattern*

Remove all variables whose names match the specified *pattern* from the environment. Also refer to the **setenv** built-in shell command, above, and the **printenv**(1) command.

ver *systype* [*command*]

With no arguments, return the current value of the SYSTYPE environment variable. With a *systype* argument, change the SYSTYPE environment variable to either **bsd4.3** or **sys5.3**, depending on which is specified.

wait Wait for all background jobs. If the shell is interactive, an interrupt can disrupt the wait, and the shell prints the names and job numbers of all jobs known to be outstanding.

which Identify which file would be executed if the command were submitted for execution. The command is submitted to normal alias and variable substitutions.

while (*expr*)

...
end While the specified expression evaluates to nonzero, evaluate the commands between the **while** and the matching **end**. You can use **break** and **continue** to terminate or continue the loop prematurely. (The **while** and **end** must appear alone on their input lines.) Prompting occurs the first time through the loop, as for the **foreach** statement if the input is a terminal.

%job Bring the specified *job* number into the foreground.

%job &
Continue the specified *job* in the background.

@ Print the values of all the shell variables.

@ name = expr
Set the specified *name* to the value of *expr*. If the expression contains a **>**, **<**, **&**, or **|** character, you must enclose at least that part in parentheses.

@ name[index] = expr
Assign the value of *expr* to the *index*th argument of *name*. Both *name* and its *index*th component must already exist.

The operators ***=**, **+=**, etc., are available as in C. The space separating the name from the assignment operator is optional. However, spaces are mandatory in separating components of *expr* that would otherwise be single words. Special postfix **++** and **--** operators increment and decrement *name*, respectively, for example, **@ i++**.

PREDEFINED AND ENVIRONMENT VARIABLES

The following variables have special meaning to **cs****h**. Of these, the shell sets **argv**, **cwd**, **home**, **path**, **prompt**, **shell**, and **status**. Except for **cwd** and **status**, this setting occurs only at initialization. These variables will not then be modified unless you explicitly perform the modification.

cs**h** copies the USER environment variable into the **user** variable; TERM into **term**; and HOME into **home**. It then copies these back into the environment whenever the normal shell variables are reset.

cs**h** handles the PATH environment variable in a similar manner. Do not worry about the setting for PATH other than in the file **.cshrc**. Inferior **cs****h** processes import the definition of **path** from the environment, and re-export it if you then change it.

argv Set to the arguments to the shell. It is from this variable that positional parameters are substituted, that is, **\$argv[1]** replaces **\$1**, etc.

cdpath Give a list of alternate directories searched to find subdirectories in **chdir** commands.

cwd Give the full pathname of the current directory.

echo	Echo each command and its arguments just before the command is executed. This variable is set when you specify the -x command line option. For non-built-in commands all expansions occur before echoing. Echo built-in commands before command and filename substitution, since these substitutions are then done selectively.
filec	Enable filename completion.
histchars	Change the characters used in history substitution, if you specify a string value. Use the first character of its value as the history substitution character, replacing the default character ! . The second character of its value replaces the ^ character in quick substitutions.
history	Control the size of the history list. If you specify a numeric value, do not discard any command that has been referenced in that many events. The last executed command is always saved on the history list. The shell may run out of memory if the value of history is too large.
home	Represents the home directory of the invoker, initialized from the environment. The filename expansion of ~ refers to this variable.
ignoreeof	If set, ignore the end-of-file from terminal input devices. This prevents shells from accidentally being killed by an EOF.
mail	Represent the files where the shell checks for mail. This is done after each command completion that results in a prompt, if a specified interval has elapsed. The shell will tell you that you have new mail, if the file exists with an access time not greater than its modify time. If the first word of the value of mail is numeric, it specifies a different mail-checking interval (in seconds) than the default (10 minutes). If you specify multiple mail files, the shell tells you that you have new mail in <i>name</i> , when there is mail in the file <i>name</i> .
noclobber	Restrict output redirection to ensure that files are not accidentally destroyed, and that redirections done with >> refer to existing files.
noglob	If set, inhibit filename expansion. Use this in shell scripts that do not deal with filenames, or after you have obtained a list of filename,s and further expansions are not desirable.
nonomatch	If set, it is not an error for a filename expansion not to match any existing files; rather, the primitive pattern is returned. It is still an error for the primitive pattern to be malformed, that is, echo [still gives an error.
notify	If set, notify the user asynchronously of job completions. By default, the shell presents job completions just before printing a prompt.

path	Each word of the path variable specifies a directory in which commands are to be sought for execution. A null word specifies the current directory. If there is no path variable, cs h executes only full pathnames. The default search path in Domain/OS BSD is (. /usr/ucb /bin /usr/bin /usr/apollo/bin) . However, this may vary from system to system. For the super-user, the default search path is (/etc /bin /usr/bin /usr/apollo/bin) , which may also vary. A shell that is given neither the -c nor the -t option normally hashes the contents of the directories in the path variable after reading .cshrc , and each time the path variable is reset. If new commands are added to these directories while the shell is active, it may be necessary to give the rehash command, or the new commands may not be found.
prompt	The string printed on the cs h command line, before the shell reads commands from an interactive terminal input. If ! appears in the string, replace it by the current event number (unless a preceding backslash is given). The default prompt is "%" ; for the super-user, the default prompt is "#" .
savehist	Give a numeric value to control the number of history list entries saved in %.history at log-out time. Save any command that has been referenced in that many events. During start-up, the shell sources %.history into the history list, enabling history to be saved across log-ins. If the value of savehist is too large, the shell is slow during start-up.
shell	Represent the file in which the shell resides. This is used in forking shells to interpret files that have execute bits set, but are not executable by the system. (See the description of Non-Built-in Command Execution, below.) This variable is initialized to the (system-dependent) home of the shell.
status	Give the status returned by the last command. If it terminated abnormally, add 0200 to the status. Built-in commands that fail return exit status 1. All other built-in commands set status 0.
time	Control automatic timing of commands, if a numeric value is supplied. If set, print the user, system, a utilization percentage, and real times for any command that takes more than this many CPU seconds, when the command terminates. A utilization percentage is the ratio of user time plus system time to real time.
verbose	Print the words of each command after history substitution. This variable is set by the -v command-line option to cs h.

NON-BUILT-IN COMMAND EXECUTION

When a command to be executed is found to be something other than a built-in command, **cs**h attempts to execute it via **execve(2)**. Each word in the variable **path** names a directory from which the shell attempts to execute the command. If you do not specify either a **-c** or a **-t** option, the shell hashes the names in these directories

into an internal table so that it tries an `exec` in a directory only if the command potentially resides there. This greatly speeds command location when a lot of directories are present in the search path. For each directory component of **path** that does not begin with a `/`, the shell concatenates with the given command name to form a path-name of a file that it then attempts to execute. The shell also does this if the internal hash table mechanism is turned off (via `unhash`), or a `-c` or `-t` command-line option is specified in `cs`.

Commands in parentheses are always executed in a sub-shell. Thus, `(cd ; pwd)` ; `pwd` prints the home directory, leaving you where you were (printing this after the home directory). On the other hand, `cd ; pwd` leaves you in the *home* directory. Commands in parentheses are most often used to prevent `chdir` from affecting the current shell.

If a file has execute permissions but is not an executable binary to the system, `cs` assumes it to be a file containing shell commands, and spawns a new shell to read it.

If there is an **alias** for **shell**, the words of the alias are prefixed to the argument list to form the shell command. The first word of the **alias** should be the full pathname of the shell (for example, `$shell`). Note that this is a special, late-occurring, case of **alias** substitution, and it only allows words to be prefixed to the argument list without modification.

COMMAND LINE OPTIONS

- b This flag forces a break from option processing, causing any further shell arguments to be treated as non-option arguments. The remaining arguments are not interpreted as shell options. You can use the `-b` to pass options to a shell script without confusion or possible subterfuge. The shell will not run a set-user ID script without this option.
- c Commands are read from the (single) following argument, which must be present. Any remaining arguments are placed in *argv*.
- D*name=value*
Set the parameter *name* to *value*, then pass it to the shell's environment. This option is useful for tailoring the environment of a shell invoked from a program that isn't a shell (such as the DM). For example, if your key definition sets the `-D` variable as follows: `kd cp /bin/csh -DNEWPAD=true ke`, the `.cshrc` script can use the value of the `NEWPAD` variable to execute additional commands or perform special processing. You can specify a number of `-D` options.
- e The shell exits if any invoked command terminates abnormally or yields a nonzero exit status.
- f The shell starts faster, because it neither searches for nor executes commands from the file `.cshrc` in the invoker's home directory.
- i The shell is interactive and prompts for its top-level input, even if it appears not to be a terminal. Shells are interactive without this option if their inputs and outputs are terminals.

- n Parse commands, but do not execute them. This aids in syntactic checking of shell scripts.
- s Take command input from the standard input.
- t Read and execute a single line of input. Use a backslash (\) to escape the new-line at the end of this line and continue onto another line.
- v Set the **verbose** variable, causing command input to be echoed after history substitution.
- x Set the **echo** variable, so that commands are echoed immediately before execution.
- V Set the **verbose** variable even before **.cshrc** is executed.
- X Set the **echo** variable even before **.cshrc** is executed.

ARGUMENT LIST PROCESSING

If argument 0 to the shell starts with a dash (-), this is a log-in shell. If arguments remain after command-line options are processed, but you did not specify one of the -c, -i, -s, or -t options, the first argument is taken as the name of a file of commands to be executed. The shell opens this file, and saves its name for possible resubstitution by \$0. Since many systems use either the standard version 6 or version 7 shells whose shell scripts are not compatible with this shell, **csh** executes a standard shell if the first character of a script is not a pound sign (#), that is, if the script does not start with a comment. Remaining arguments initialize the **argv** variable.

SIGNAL HANDLING

csh normally ignores **quit** signals. Jobs running detached, either by **&**, or the **bg** or **%... &** commands, are immune to signals generated from the keyboard, including hangups. Other signals have the values the shell inherited from its parent. Use **onintr.** to control the shell's handling of interrupts and terminate signals in shell scripts. Log-in shells catch the **TERM** (terminate) signal. Otherwise, this signal is passed on to children from the state in the shell's parent. In no case are interrupts allowed when a log-in shell is reading the file **.logout**.

LIMITATIONS

Words can be no longer than 1024 characters. The system limits argument lists to 10240 characters. The number of arguments to a command that involves filename expansion is limited to one-sixth the number of characters allowed in an argument list. Command substitutions may substitute no more characters than are allowed in an argument list. To detect looping, the shell restricts the number of **alias** substitutions on a single line to 20.

WARNINGS

csh will not read **.cshrc** or **.login** if the files are owned by another user.

BUGS

When a command is restarted from a stop, the shell prints the directory it started in if this is different from the current directory; this can be misleading (that is, wrong) as the job may have changed directories internally.

Shell built-in functions are not stoppable/restartable. Command sequences of the form 'a ; b ; c' are also not handled gracefully when stopping is attempted. If you suspend 'b', the shell immediately executes 'c'. This is especially noticeable if this expansion results from an alias. It suffices to enclose the sequence of commands in parentheses to force it to a subshell, for example, '(a ; b ; c)'.

Control over tty output after processes are started is primitive; perhaps this will inspire someone to work on a good virtual terminal interface. In a virtual terminal interface much more interesting things could be done with output control.

Alias substitution is most often used to clumsily simulate shell procedures; you should use shell procedures rather than aliases.

Commands within loops, prompted for by "?", are not placed in the history list. Control structure should be parsed rather than being recognized as built-in commands. This would allow control commands to be placed anywhere, to be combined with |, and to be used with & and ; metasyntax.

It should be possible to use the : modifiers on the output of command substitutions. All and more than one : modifier should be allowed on \$ substitutions.

The way the filec facility is implemented is ugly and expensive.

The built-in time function correctly calculates user time, system time, elapsed time, and utilization percentages. The number of input and output requests, page faults, dirty pages, reclaimed pages, as well as the amount of shared/unshared/total memory may be calculated in error. In most cases, the data displayed is simply 0.

FILES

~/cshrc	Read at beginning of execution by each shell.
~/login	Read by log-in shell, after .cshrc at login.
~/logout	Read by log-in shell, at logout.
/bin/sh	Standard shell, for shell scripts not starting with a '#'.
/tmp/sh*	Temporary file for '<<<'.
/etc/passwd	Source of home directories for '~name'.

SEE ALSO

sh(1), access(2), execve(2), fork(2), killpg(2), pipe(2), setrlimit(2), sigvec(2), umask(2), wait(2), tty(4), a.out(5), environ(7);
Using Your BSD Environment.

NAME

ctags – create a tags file

SYNOPSIS

ctags [**-BFatuwx**] [**-f tagfile**] *name* ...

DESCRIPTION

ctags makes a tags file for **ex/vi**(1) from the specified C, Pascal, FORTRAN, **yacc**(1), **lex**(1), and LISP sources. A tags file gives the locations of specified objects (in this case functions and typedefs) in a group of files. Each line of the tags file contains the object name, the file in which it is defined, and an address specification for the object definition. Functions are searched with a pattern, typedefs with a line number. Specifiers are given in separate fields on the line, separated by blanks or tabs. Using the tags file, **ex** can quickly find these object's definitions.

Normally **ctags** places the tag descriptions in a file called **tags**; this may be overridden with the **-f** option.

Files whose names end in **.c** or **.h** are assumed to be C source files and are searched for C routine and macro definitions. Files whose names end in **.y** are assumed to be YACC source files. Files whose names end in **.l** are assumed to be either LISP files if their first non-blank character is **;**, **(**, or **[**, or lex files otherwise. Other files are first examined to see if they contain any Pascal or FORTRAN routine definitions; if not, they are processed again looking for C definitions.

The tag **main** is treated specially in C programs. The tag formed is created by prepending **M** to the name of the file, with a trailing **.c** removed, if any, and leading pathname components also removed. This makes use of **ctags** practical in directories with more than one program.

OPTIONS

- B** Use backward searching patterns (**?...?**).
- F** Use forward searching patterns (**/.../**) (default).
- a** Append to tags file.
- t** Create tags for typedefs.
- u** Update the specified files in tags, that is, all references to them are deleted, and the new values are appended to the file. (Beware: this option is implemented in a way which is rather slow; it is usually faster to simply rebuild the tags file.)
- w** Suppress warning diagnostics.
- x** Produce a list of object names, the line number and filename on which each is defined, as well as the text of that line, and prints this on the standard output. This is a simple index that you can print out as an off-line readable function index.
- v** Produce an index of the form expected by **vgrind**(1), on the standard output. This listing contains the function name, filename, and page number (assuming 64-line pages). Since the output is sorted in lexicographic order, you may wish to run the output

through **sort -f**. Sample use:

```
ctags -v files | sort -f > index  
vgrind -x index
```

-f *tagfile* Place the tag descriptions in the specified *tagfile* instead of the file named **tags**.

BUGS

Recognition of **functions**, **subroutines** and **procedures** for FORTRAN and Pascal is done in a very simple-minded way. No attempt is made to deal with block structure; if you have two Pascal procedures in different blocks with the same name, you lose.

The method of deciding whether to look for C or Pascal and FORTRAN functions is a hack.

Does not know about **#ifdefs**.

Should know about Pascal types. Relies on the input being well formed to detect typedefs. Use of **-tx** shows only the last line of typedefs.

FILES

tags Output tags file

SEE ALSO

ex(1), **vgrind(1)**, **vi(1)**

NAME

cvtname – convert pathnames between upper and lowercase and preserve colons

SYNOPSIS

cvtname [*options*]

DESCRIPTION

Prior to SR10, the colon (:) was used as an escape character for the purpose of storing mixed-case names. For example, the filename "Readme" was stored as ":readme". Domain/OS programs mapped ":r" and interpreted it as "R". In pre-SR10 Aegis-only environments, colons used in pathnames were treated as literal characters, since Aegis was not case sensitive.

Colon-character constructs in pathnames from pre-SR10 file systems are converted to the appropriate uppercase letter (or special character) automatically when they are copied to SR10 systems. The **cvtname** command allows you to selectively undo that process and thereby restore literal colons to pathnames. **cvtname** also allows you to convert pathnames to all uppercase or all lowercase. The tool operates on entire pathnames. That is, you cannot convert one capital letter in an SR10 pathname back to a "colon-character" sequence without converting them all.

Regardless of the mode specified, **cvtname** queries you before converting each pathname, unless you specify **-nq**, in which case the changes are applied to all objects subordinate to the pathname specified.

OPTIONS

Without options, **cvtname** converts capital letters back to colon-character sequences.

- | | |
|---------------------------|---|
| -m <i>pathname</i> | Convert capital letters in the names of all objects in <i>pathname</i> back to colon-character sequences. If -li is also specified, potential changes are listed but no changes are made. If -nq is present, the changes are done automatically and all modified names are listed. (The default is -m without -nq) |
| -l <i>pathname</i> | Convert <i>pathname</i> and subordinate object names to all lowercase. If -li is also specified, potential changes are listed but no changes are made. If -nq is present, the changes are done automatically and all modified names are listed. |
| -u <i>pathname</i> | Convert <i>pathname</i> and subordinate object names to all uppercase. If -li is also specified, potential changes are listed but no changes are made. If -nq is present, the changes are done automatically and all modified names are listed. |

EXAMPLES

The following example allows you to convert the capital letters or colon-character constructs in pathnames in the directory **leduc**, querying you before making any changes. Output is shown under the command line. The left-hand column shows unconverted name; right shows converted. Type **y** to convert, **n** to keep old name.

\$ cvtname /ledu

/ledu/:C	/ledu/:::c n
/ledu/CAT	/ledu/:c:a:t y
/ledu/CAT converted to	/ledu/:c:a:t

The following example allows you to selectively convert pathnames in **ledu** to upper-case.

\$ cvtname -upper /ledu

/ledu	/LEDU n
/ledu/:c	/ledu/:C n
/ledu/:c:a:t	/ledu/:C:A:T n
/ledu/acl_from_whoville	/ledu/ACL_FROM_WHOVILLE y
/ledu/acl_from_whoville converted to	/ledu/ACL_FROM_WHOVILLE
/ledu/backup.pas	/ledu/BACKUP.PAS n
/ledu/ffl	/ledu/FF1 y
/ledu/ffl converted to	/ledu/FF1
/ledu/TD/backup_history	/ledu/TD/BACKUP_HISTORY n

NAME

cvtrgy – convert registry between SR9.x and SR10 formats

SYNOPSIS

cvtrgy [**–from9to10** | **–from10to9** [**–invalidate_unixids**]] **–readonly** |
–owner pgo | **–first** | **–nq** | **–from source_rgy** **–to dest_rgy**

DESCRIPTION

The **cvtrgy** command allows the system administrator to generate an SR10 format registry database from SR9.7 registry files, or generates SR9.7 registry files with data from the SR10 registry. The tool operates on SR9.7 nodes only. Both the **rgyd** and **llbd** servers must be running on the SR10 node, except when the **–first** option is used. Run **cvtrgy** the first time when you add SR10 nodes to your network, and periodically thereafter to synchronize the pre-SR10 and SR10 registry information.

You must specify either **–from9to10** or **–from10to9**. By default, **cvtrgy** creates a read-only registry of the destination type. That is, **cvtrgy –from9to10** creates a read-only SR10 format master registry, while **cvtrgy –from10to9** creates a read-only SR9.x format master registry. You can then propagate the information to replica registries in the appropriate way.

Whenever the conversion from SR10 to SR9 occurs, if the registry files exist at the destination node specified in the command line, the tool quits without updating. This means that before running **cvtrgy –from10to9**, you should rename (or move) the SR9.x registry database on the destination node.

The **cvtrgy** tool assigns UNIX identifiers automatically during the conversion process if you prefer. However, if your pre-SR10 node runs Domain/OS, you should preserve the identifiers associated with accounts in your current (pre-SR10) **/etc/passwd** and **/etc/group** files. In normal operation, **cvtrgy** looks for the **/etc/passwd** and **/etc/group** files and assigns identifiers from them, if they exist. Therefore, you should run **cvtrgy** on a 9.7 node that either contains your master **/etc/passwd** and **/etc/group** files or has a link to them.

If **cvtrgy** doesn't find the **/etc/passwd** and **/etc/group** files and an **/etc** directory exists, it queries you before assigning new UNIX identifiers, unless the **–nq** (no query) flag is turned on, in which case **cvtrgy** exits with an error.

In order to add or change accounts and other registry data, you must edit the writable registry with the tool appropriate to the registry's format (i.e., with **edrgy** on SR10, **edacct** and **edppo** on SR9.x) on a node running the same software release as the format of the writable registry. Thus, if your SR9.x registries were writable, you'd have to edit them using **edacct** and **edppo**, from a node running SR9.7. Once your SR10 registry is the writable one, use **edrgy**.

The **cvtrgy** tool resides in the **/install/tools/cvtrgy** after an SR10 installation and must be copied to an SR9.7 node before you run it. After running **cvtrgy**, you must also run the **crpasswd** command on an SR9.x node to update the **/etc/passwd** and **/etc/group** files. The SR10 directory **/install/tools** contains a new version of **crpasswd** which you should copy to all SR9.7 nodes that need to run **crpasswd**. (You can rename or replace the old version of **crpasswd**.) See "Making the Transition to SR10 Operating System Releases" for further details on running **cvtrgy**.

OPTIONS

- from9to10** Convert SR9.x registry files to SR10 registry format
- from10to9** Convert SR10 registry data to SR9.7 format and place in SR9.7 registry files
- from *source_rgy*** Specify source for registry data to be converted. For **-from9to10**, must be in the form *//node_name/registry/rgy_site*. For **-from10to9**, must be *//node_name*. Either or both registry sites may be remote from the node running **cvtrgy**.
- to *dest_rgy*** Specify destination for converted registry data. For **-from9to10**, must be in the form *//node_name/registry/rgy_site*. For **-from10to9**, must be *//node_name*. Either or both registry sites may be remote from the node running **cvtrgy**.
- owner *pgo*** Specify SR10 registry owner, in the SID form *p.g.o*, where all *pgo* names and the *pgo* account already exist in the SR9.7 registry. *pgo* is a string of the form *pers.group.org*. You must specify with every invocation of **-from9to10**. This option is meaningful only with the **-from9to10** option.
- first** Specify that this is the first invocation of **cvtrgy**. In this case only, **cvtrgy** runs without **rgyd** and **llbd** servers running. Use only once. Only meaningful with **-from9to10**.
- readonly** Make SR9.7 registries read-only, permanently. Only meaningful with **-from9to10**. Can only be run in this mode once; after running, cannot use **-from9to10** again.
- nq** No query. Silent mode. Don't query before assigning new UNIX identifiers (**cvtrgy** quits). Don't query for owner (**cvtrgy** quits).
- invalidate_unixids** If you've edited UNIX IDs (numbers) in the SR9.7 */etc/passwd* or */etc/group* after you've already run **cvtrgy** at least once, you should propagate the new numbers to the SR10 registry. Running **cvtrgy** with this option, in the **-from9to10** direction, invalidates the UNIX IDs stored within the SR9.7 registry files. After running **cvtrgy** with this option, you then need to rerun **crpasswd** followed by **cvtrgy -from9to10** as you normally would. Finally, you must also run */etc/syncids* on all SR10 disks. Only meaningful with **-from9to10**.

CONVERTING FROM SR9.7 TO SR10

You must be root to run **cvtrgy**. Use the following command line. The **node_name1** is the SR9.7 node.

```
$ cvtrgy -from9to10 -from //node_name1/registry/rgy_site
      -to //node_name2 -owner pgo -first
```

CONVERTING FROM SR10 TO SR9.7

The person who runs the tool must be logged in as root or locksmith. Use the following command line. The **node_name1** is the SR10 node.

```
$ cvtrgy -from10to9 -from //node_name1 -to //node_name2/registry/rgy_site
```

EXAMPLE

The following is a sample transcript from a **cvtrgy** session that converts SR9.x registry data files to an SR10 format registry database. This is the first time **cvtrgy** has been run on the network. A single collision is shown to illustrate **cvtrgy**'s warning message format; you may see more warnings at your site.

```
$ cvtrgy -from9to10 -from //dog/registry/rgy_site1 -to //cat -first -owner %.sys_admin.%
```

Phase 1 - opening registry files:

Phase 2 - modifying SR9 registry files:

```
Converted person file saved in registry
//dog/registry/rgy_site1
```

```
Converted project file saved in registry
//dog/registry/rgy_site1
```

```
Converted org file saved in registry
//dog/registry/rgy_site1
```

Phase 3 - converting person file:

```
?(cvtrgy) Warning - unix id collision:
person bin_sr9 reassigned from 3 to 10002
Converted person file saved in registry
//dog/registry/rgy_site1
```

Phase 4 - converting project file:

```
?(cvtrgy) Warning - unix id collision:
project backup reassigned from 1001 to 3
Converted project file saved in registry
//dog/registry/rgy_sitel
```

Phase 5 - converting org file:

```
Converted org file saved in registry
//dog/registry/rgy_sitel
```

Phase 6 - converting accounts:

Phase 7 - adding default accounts:

```
Converted account file saved in registry
//dog/registry/rgy_sitel
```

Phase 8 - closing the sr9 registry files:

Phase 9 - writing conversions to sr10 registry:

Conversion completed successfully:

SEE ALSO

passwd(5), group(5)

NAME

cvt_font – convert fonts from pre-SR10 to SR10 format

SYNOPSIS

cvt_font *destination source1* [*source2*]

DESCRIPTION

The **cvt_font** command creates a new font file formatted for SR10. If one *source* name is given, it is converted and placed in the *destination* file. If two source names are given, then the characters in the second source font are concatenated with the characters in the first font, converted, then placed in the *destination* font file.

The source font(s) must be in pre-SR10 format. Since all pre-SR10 fonts have space pre-allocated for 128 characters, the new font can contain up to 256 characters.

If the *destination* font file already exists, or if **cvt_font** fails to find either *source* file, an error is printed, and the command terminates without changing any fonts.

EXAMPLES

The following example takes the **vt100s** font from **/sys/dm/fonts** and formats it for SR10 in the file **vt100s** in the working directory:

```
$ cvt_font vt100s /sys/dm/fonts/vt100s
```

The following example takes the **courier10** and **courier10.a** font files from **/sys/dm/fonts**, concatenates them, and formats them for SR10 in the file **courier10** in the working directory:

```
$ cvt_font courier10 /sys/dm/fonts/courier10 /sys/dm/fonts/courier10.a
```

SEE ALSO

tr_font(1)

NAME

date – print and set the date

SYNOPSIS

date [**-n**] [**-u**] [*yyymmddhhmm*[*.ss*]]

DESCRIPTION

If you don't specify an argument, **date** prints the current date and time. Providing an argument sets the desired date. Only the super-user can set the date.

OPTIONS

-u Display or set the date in GMT (universal time).

yyymmddhhmm[*.ss*]

If you're setting the date, *yy* represents the last two digits of the year; the first *mm* is the month number; *dd* is the day number; *hh* is the hour number (24-hour system); the second *mm* is the minute number; *.ss* is optional and represents the seconds.

For example:

date 8906131627

sets the date to June 13 1989, 4:27 PM. The year, month and day can be omitted; the default values will be the current ones. The system operates in GMT. The **date** command takes care of the conversion to and from local standard and daylight-saving time.

-n If **timed** is running to synchronize the clocks of machines in a local area network, **date** sets the time globally on all those machines, unless you specify the **-n** option.

DIAGNOSTICS

Exit status is 0 on success; 1 on complete failure to set the date; 2 on successfully setting the local date but failing globally.

FILES

/usr/adm/wtmp
/usr/adm/messages

To record time-setting.

To record the name of the user setting the time.

WARNING

Using **date** to set current date and time backwards can cause the creation of objects possessing previously assigned UIDs.

At boot-time, the Domain/OS incorporates the current date and time into the algorithm by which it computes UIDs (unique identifiers). After using **date** to set the system time backwards, the user must not reboot the machine until the previous system time has been passed. Rebooting prior to this time will incur the risk that the UID algorithm will incorporate a previously used date and time into new UIDs. As Domain/OS relies on the uniqueness of UIDs, this can have devastating results.

SEE ALSO

gettimeofday(2), **utmp(5)**

NAME

dbacl – Domain/Dialogue–based access control list editor

SYNOPSIS

dbacl [*file*]

DESCRIPTION

dbacl provides an interactive menu-based editor for manipulating Access Control Lists (ACLs). It is primarily designed with novice or occasional ACL users in mind. **chacl**(1), **cpacl**(1), and **lsacl**(1) are better suited to complicated actions on large numbers of ACLs.

To use **dbacl**, press the left mouse key (or the F1 key) to select items on the screen such as buttons or ACL entries. The bar at the top of the screen contains a "Quit" button, and names of menus. By pressing the mouse key over one of the menu names, you are presented with a pop-up menu with commands. Pull down and release the key over the name of a command to select it.

If a command appears in grayed-out text, it is not currently selectable. For example, before selecting "Cut" from the "Entry" menu, you must select an entry to cut, by clicking on it with the mouse.

You can use the right mouse key (or the F3 key) as a short cut for selecting an entry and choosing the "Change Entry" command.

If a button has a double outline, you can select it by pressing the RETURN key anywhere in its window. Likewise, the ESC key nearly always cancels the current command.

SEE ALSO

chacl(1), **cpacl**(1), **lsacl**(1), **acl**(7), **salacl**(8)

NAME

dbx – source code debugger

SYNOPSIS

```
dbx [ -r ] [ -i ] [ -k ] [ -I dir ] [ -no_src ] [ -no_frame ] [ -c file ]
[ objfile ]
```

DESCRIPTION

dbx is a tool for source-level debugging and execution of programs under BSD. The *objfile* is an object file produced by a compiler, with the appropriate flag (usually **-g**) specified to produce symbol information in the object file. Currently, **cc(1)**, and **f77(1)** produce the appropriate source information. You can use the machine-level facilities of **dbx** on any program. All compilers and assemblers produce debugging information usable by **dbx**.

The object file contains a symbol table that includes the name of the all the source files translated by the compiler to create it. These files are available for perusal while you use the debugger.

If the file **.dbxinit** exists in the current directory, the debugger commands in it are executed. **dbx** also checks for a **.dbxinit** in your home directory if there isn't one in the current directory.

dbx creates a separate transcript pad for debugger interactions unless you specify the **-no_frame** option. **dbx** also creates a window to display source code unless you specify the **-no_src**.

OPTIONS

- r** Execute *objfile* immediately. If it terminates successfully, **dbx** exits. Otherwise the reason for termination is reported and you are offered the choice of entering the debugger or letting the program fault. **dbx** reads from **/dev/tty** when you specify **-r** and standard input is not a terminal.
 - i** Force **dbx** to act as though standard input is a terminal.
 - k** Map memory addresses. This option is useful for kernel debugging.
 - I dir** Add *dir* to the list of directories that are searched when looking for a source file. Normally **dbx** looks for source files in the current directory and in the directory where *objfile* is located. You can also use the **use** command to set the directory search path.
 - c file** Execute the **dbx** commands in the *file* before reading from standard input.
 - no_src** Disable source display.
 - no_frame** Do not create a separate debugger transcript pad.
- Unless you specify **-r**, **dbx** just prompts and waits for a command.

Execution and Tracing Commands

run [*args*] [< *filename*] [> *filename*]

rerun [*args*] [< *filename*] [> *filename*]

Start executing *objfile*, passing *args* as command line arguments; you can use < or > to redirect input or output in the usual manner. When you use **rerun** without any arguments, the previous argument list is passed to the program; otherwise it is identical to **run**. If *objfile* was written since the last time the symbolic information was read in, **dbx** reads in the new information.

trace [*in procedure/function*] [*if condition*]

trace *source-line-number* [*if condition*]

trace *procedure/function* [*in procedure/function*] [*if condition*]

trace *expression at source-line-number* [*if condition*]

trace *variable* [*in procedure/function*] [*if condition*]

Have tracing information printed when the program is executed. A number is associated with the command that is used to turn the tracing off (see the **delete** command).

The first argument describes what is to be traced. If it is a *source-line-number*, the line is printed immediately before being executed. You must precede source-line numbers in a file other than the current one by the name of the file in quotation marks and a colon, for example, "**mumble.p**":17.

If the argument is a procedure or function name, then every time it is called, information is printed telling what routine called it, from what source line it was called, and what parameters were passed to it. In addition, its return is noted, and if it's a function, the value it is returning is also printed.

If the argument is an *expression* with an **at** clause, the value of the expression is printed whenever the identified source line is reached.

If the argument is a variable, the name and value of the variable is printed whenever it changes. Execution is substantially slower during this form of tracing.

If no argument is specified, all source lines are printed before they are executed. Execution is substantially slower during this form of tracing.

The clause "**in procedure/function**" specifies that tracing information is printed only while **dbx** is executing inside the given procedure or function.

condition is a Boolean expression and **dbx** evaluates it before it prints the tracing information; if *condition* is false, **dbx** does not print the information.

stop if condition

stop at source-line-number [if condition]

stop in procedure/function [if condition]

stop variable [if condition]

Stop execution when the given line is reached; procedure or function called; variable changed; or condition true.

status [> filename]

Print the currently active **trace** and **stop** commands.

delete command-number ...

Remove the traces or stops corresponding to the given numbers. The **status** command prints the numbers associated with traces and stops.

catch number

catch signal-name

ignore number

ignore signal-name

Start or stop trapping a signal before it is sent to the program. This is useful when a program being debugged handles signals such as interrupts. You can specify a signal by number or by a name (for example, SIGINT). Signal names are case-insensitive and the "SIG" prefix is optional. By default all signals are trapped except SIGCONT, SIGCHLD, SIGALRM and SIGKILL.

cont integer

cont signal-name

Continue execution from where it stopped. If a signal is specified, the process continues as though it received the signal. Otherwise, the process is continued as though it had not been stopped.

You cannot continue to execute a process if it has "finished", that is, called the standard procedure **exit()**. **dbx** does not allow the process to exit, thereby letting you examine the program state.

step Execute one source line.

next Execute up to the next source line. The difference between this and **step** is that if the line contains a call to a procedure or function the **step** command stops at the beginning of that block, while the **next** command does not.

return [procedure]

Continue until a return to *procedure* is executed, or, if you do not specify a procedure, until the current procedure returns.

call procedure(parameters)

Execute the object code associated with the named procedure or function.

Printing Variables and Expressions

Names are resolved first using the static scope of the current function, then using the dynamic scope if the name is not defined in the static scope. If static and dynamic searches do not yield a result, an arbitrary symbol is chosen and the message "[using *qualified name*]" is printed. The name-resolution procedure may be overridden by qualifying an identifier with a block name, for example, "*module.variable*". For C, source files are treated as modules named by the filename without ".c".

Expressions are specified with an approximately common subset of C and Pascal (or equivalently Modula-2) syntax. Indirection can be denoted using either a prefix "*" or a postfix "~" and array expressions are subscripted by brackets ("[]"). The field reference operator (".") can be used with pointers as well as records, making the C operator "->" unnecessary (although it is supported).

dbx checks types of expressions; use "*type-name(expression)*" to override the type of an expression. When there is no corresponding named type, you can use the special constructs "&*type-name*" and "\$*tag-name*" to represent a pointer to a named type or C-structure tag.

assign *variable* = *expression*

Assign the value of the expression to the variable.

dump [*procedure*] [> *filename*]

Print the names and values of variables in the given procedure, or the current procedure if you did not specify one. If you specify ".", for procedure, dbx dumps all active variables.

print *expression* [, *expression* ...]

Print out the values of the expressions.

whatis *name*

Print the declaration of the given name, which you can qualify with block names as above.

which *identifier*

Print the full qualification of the given identifier, that is, the outer blocks that the identifier is associated with.

up [*count*]

down [*count*]

Move the current function, which is used for resolving names, up or down the stack *count* levels. The default *count* is 1.

where Print a list of the active procedures and function.

whereis *identifier*

Print the full qualification of all the symbols whose name matches the given identifier. The order in which the symbols are printed is not meaningful.

Accessing Source Files

/regular expression[/]

?regular expression[?]

Search forward or backward in the current source file for the given pattern.

edit [*filename*]

edit *procedure/function-name*

Invoke an editor on *filename* or the current source file if you did not specify one. If you specify a *procedure* or *function* name, **dbx** invokes the editor on the file that contains it. Which editor is invoked by default depends on the installation. Set the environment variable EDITOR to the name of the desired editor to override the default.

file [*filename*]

Change the current source filename to *filename*. If you do not specify one, **dbx** prints the current source filename.

func [*procedure/function*]

Change the current function. If you do not specify one, **dbx** prints the current function. Changing the current function implicitly changes the current source file to the one that contains the function; it also changes the current scope used for name resolution.

list [*source-line-number* [, *source-line-number*]]

list *procedure/function*

List the lines in the current source file from the first line number to the second, inclusive. If no lines are specified, **dbx** lists the next 10 lines. If the name of a procedure or function is given, **dbx** lists lines *n-k* to *n+k*, where *n* is the first statement in the procedure or function and *k* is small.

use *directory-list*

Set the list of directories to be searched when looking for source files.

Command Aliases and Variables

alias *name name*

alias *name* "*string*"

alias *name (parameters)* "*string*"

When commands are processed, **dbx** first checks to see if the word is an alias for either a command or a string. If it is an alias, **dbx** treats the input as though the corresponding string (with values substituted for any parameters) were entered. For example, to define an alias **rr** for the command **rerun**, you can specify

alias rr rerun

To define an alias **b** that sets a stop at a particular line you can specify

alias b(x) "stop at x"

Subsequently, the command **b(12)** expands to **stop at 12**.

set *name* [= *expression*]

The **set** command defines values for debugger variables. The names of these variables cannot conflict with names in the program being debugged, and are expanded to the corresponding expression within other commands. The following variables have a special meaning:

\$frame

Setting this variable to an address causes **dbx** to use the stack frame pointed to by the address for doing stack traces and accessing local variables. (This variable is not supported by Domain/OS BSD).

\$hexchars

\$hexints

\$hexoffsets

\$hexstrings

When these variables are set, **dbx** prints characters, integers, offsets from registers, or character pointers, respectively, in hexadecimal.

\$listwindow

The value of this variable specifies the number of lines to list around a function or when you specify the **list** command without any parameters. This value is also used when displaying source in the source window. The current line is positioned so that as much of the listwindow as possible is visible. Its default value is 10.

\$mapaddrs

Setting (unsetting) this variable causes **dbx** to start (stop) mapping addresses. As with **\$frame**, this is useful for kernel debugging.

\$unsafecall

\$unsafeassign

When the **\$unsafecall** variable is set, strict type checking is turned off for arguments to subroutine or function calls (for example, in the **call** statement). When **unsafeassign** is set, strict type-checking between the two sides of an **assign** statement is turned off. These variables should be used only with great care, because they severely limit **dbx**'s usefulness for detecting errors.

unalias *name*

Remove the alias with the given *name*.

unset *name*

Delete the debugger variable associated with *name*.

Machine Level Commands

tracel [*address*] [*if cond*]

tracel [*variable*] [**at** *address*] [*if cond*]

stopi [*address*] [*if cond*]

stopi [**at**] [*address*] [*if cond*]

Turn on tracing or set a stop using a machine instruction address.

stepi

nexti Single step as in **step** or **next**, but do a single instruction rather than source line.

address *address* / [*mode*]

address / [*count*] [*mode*]

Print the contents of memory starting at the first *address* and continuing up to the second *address* or until *count* items are printed. If the address is ".", **dbx** uses the address following the one printed most recently. The *mode* specifies how memory is to be printed; if you omit it, **dbx** uses the previous mode specified. The initial mode is "X". The following modes are supported:

- i** Print the machine instruction.
- d** Print a short word in decimal.
- D** Print a long word in decimal.
- o** Print a short word in octal.
- O** Print a long word in octal.
- x** Print a short word in hexadecimal.
- X** Print a long word in hexadecimal.
- b** Print a byte in octal.
- c** Print a byte as a character.
- s** Print a string of characters terminated by a null byte.
- f** Print a single precision real number.
- g** Print a double precision real number.

Specify symbolic addresses by preceding the name with an "&". Specify **\$rn**, where *n* is the number of the register, to denote registers. Addresses can be expressions made up of other addresses and the operators "+", "-", and indirection (unary "*").

Miscellaneous Commands

gripe Invoke a mail program to send a message to the person in charge of **dbx**. (This command is not supported by Domain/OS BSD).

help Print a synopsis of **dbx** commands.

quit Exit **dbx**.

sh *command-line*

Pass the command line to the shell for execution. The SHELL environment variable determines which shell is used.

source *filename*

Read **dbx** commands from the given *filename*.

BUGS

dbx suffers from the same "multiple include" malady as did **sdb**. If you have a program consisting of a number of object files and each is built from source files that include header files, the symbolic information for the header files is replicated in each object file. Since about one debugger start-up is done for each link, having the linker (**ld**) re-organize the symbol information would not save much time, though it would reduce some of the disk space used.

This problem is an artifact of the unrestricted semantics of **#includes** in C; for example an **include** file can contain static declarations that are separate entities for each file in which they are included. However, even with Modula-2 a lot of duplication of symbol information is necessary for inter-module type checking.

Some problems remain with the support for individual languages. FORTRAN problems include the following

- Inability to assign to logical, logical*2, complex and double complex variables
- Inability to represent parameter constants that are not type integer or real
- Peculiar representation for the values of dummy procedures (the value shown for a dummy procedure is actually the first few bytes of the procedure text; to find the location of the procedure, use "&" to take the address of the variable)

FILES

a.out	object file
.dbxinit, ~/.dbxinit	initial commands

SEE ALSO

cc(1), dde(1), f77(1)

NAME

dc – desk calculator

SYNOPSIS

dc [*file*]

DESCRIPTION

dc is an arbitrary precision arithmetic package. Ordinarily, it operates on decimal integers, but you can specify an input base, output base, and a number of fractional digits to be maintained. **bc**(1) is the preprocessor to this package, providing infix notation, a C-like syntax that implements functions, and reasonable control structures for programs. **dc** is, fundamentally, a stacking calculator. If you specify an argument, **dc** takes input from that *file* until its end, and then takes it from the standard input.

OPERATORS

<i>number</i>	The value of the number pushed on the stack. A <i>number</i> is an unbroken string of the digits 0–9. Precede it by an underscore (<code>_</code>) to signify a negative number. A <i>number</i> can also contain decimal points.
<code>+ - / * % ^</code>	Add (+), subtract (–), multiply (*), divide (/), remainder (%), or exponentiate (^) the top two values on the stack. Pop the two entries off the stack; push the result on the stack in their place. Ignore any fractional part of an exponent.
sx	Pop the top of the stack and store it in a register named <i>x</i> , where <i>x</i> can be any character. If you specify capital S , treat <i>x</i> as a stack and push the value on it.
lx	Push the value in <i>x</i> on the stack. The register <i>x</i> remains unchanged. All registers start with zero value. If you specify capital L , treat <i>x</i> as a stack and pop its top value onto the main stack.
d	Duplicate the top value on the stack.
p	Print the top value on the stack. Do not change the top value.
P	Interpret the top of the stack as an ASCII string, remove the string, and then print it.
f	Print all values on the stack.
q	Exit the program. If executing a string, pop the recursion level by two.
Q	Pop the top value on the stack, and the string execution level by that value.
x	Treat the top element of the stack as a character string, and execute it as a string of dc commands.
X	Replace the number on the top of the stack with its scale factor.
[...]	Put the bracketed ASCII string onto the top of the stack.

<x >x =x	Pop the top two elements of the stack and compare them. Evaluate register <i>x</i> if the elements obey the stated relation.
v	Replace the top element on the stack by its square root. Take any existing fractional part of the argument into account, but otherwise ignore the scale factor.
!	Interpret the rest of the line as a UNIX command.
c	Pop all values on the stack.
i	Pop the top value on the stack and use it as the number radix for further input. Push the input base on top of the stack.
o	Pop the top value on the stack and use it as the number radix for further output.
O	Push the output base on top of the stack.
k	Pop the top of the stack and use that value as a non-negative scale factor. Print the appropriate number of places output, and maintain them during multiplication, division, and exponentiation. The interaction of scale factor, input base, and output base is reasonable if all are changed together.
z	Push the stack level onto the stack.
Z	Replace the number on the top of the stack with its length.
?	Take a line of input from the input source (usually the terminal) and execute it.
;;	Used by bc(1) for array operations.

EXAMPLE

To print the first 10 values of *n!*, specify the following:

```
[la1+dsa*pla10>y]sy
0sa1
lyx
```

DIAGNOSTICS

“x is unimplemented”

“Stack empty”

“Out of space”

“Out of headers”

“Out of pushdown”

“Nesting Depth”

Value of *x* is an octal number.

There are not enough elements on the stack to do what was asked.

The free list is exhausted (too many digits).

Too many numbers are being retained.

Too many items are on the stack.

There are too many levels of nested execution.

SEE ALSO

bc(1)

NAME

dd – convert and copy a file

SYNOPSIS

dd [*option=value*] ...

DESCRIPTION

The **dd** command copies the specified input file to the specified output with possible conversions. It uses the standard input and output by default. You can specify the input and output block size to take advantage of raw physical I/O. After completion, **dd** reports the number of whole and partial input and output blocks.

OPTIONS

if=file	Input filename; standard input is the default.
of=file	Output filename; standard output is the default.
ibs=n	Input block size <i>n</i> bytes; 512 is the default.
obs=n	Output block size; 512 is the default.
bs=n	Set both input and output block size, superseding ibs and obs . If you do not specify conversion, this is particularly efficient since no copy is required.
cbs=n	Conversion buffer size; used only if ascii , unblock , ebcdic , ibm , or block conversion is specified. In the first two cases, cbs characters are placed in the conversion buffer, any specified character mapping is done, trailing blanks are trimmed, and a newline is added before sending the line to the output. In the latter three cases, characters are read into the conversion buffer, and blanks are added to make up an output record of size cbs .
skip=n	Skip <i>n</i> input records before starting copy.
files=n	Copy <i>n</i> input files before terminating. This option makes sense only where input is a magtape or similar device.
seek=n	Seek <i>n</i> records from the beginning of the output file before copying.
count=n	Copy only <i>n</i> input records.
conv=ascii	Convert EBCDIC to ASCII.
ebcdic	Convert ASCII to EBCDIC.
ibm	A slightly different map of ASCII to EBCDIC.
block	Convert variable-length records to fixed-length records.
unblock	Convert fixed-length records to variable-length records.
lcase	Map alphabetic to lowercase.
ucase	Map alphabetic to uppercase.
swab	Swap every pair of bytes.
noerror	Do not stop processing on an error.
sync	Pad every input block 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 can be separated by **x** to indicate a product.

NOTES

Certain combinations of arguments to **conv=** are permitted. However, you cannot combine the **block** or **unblock** options with **ascii**, **ebcdic** or **ibm**. Invalid combinations silently ignore all but the last mutually-exclusive keyword.

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.

EXAMPLE

To read an EBCDIC tape, blocked ten 80-byte EBCDIC card images per record, into the ASCII file **x**, type the following:

```
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.

DIAGNOSTICS

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

BUGS

You must specify **conv=noerror, sync** when copying raw disks with bad sectors to ensure that **dd** stays synchronized.

Domain/OS BSD does not support some raw I/O devices typically used with **dd**.

SEE ALSO

cp(1), **tr(1)**

NAME

dde – Domain Distributed Debugging Environment

SYNOPSIS

```
dde [-do "cmd_list"
    [ [-on target_machine] [-target_type target_type]
    { [-input pathname] [-output pathname [-ao]]
      [-errors pathname [-ae]] program_invocation
    | -attach process_id } ]
```

DESCRIPTION

The **dde** command invokes the Domain Distributed Debugging Environment, the standard debugger for the Domain/OS operating system at SR10. For complete information about this debugger and its commands, consult the *Domain Distributed Debugging Environment Reference* (011024) or invoke the debugger's own **help** command for online assistance.

OPTIONS

- | | |
|--|---|
| -do " <i>cmd_list</i> " | Execute <i>cmd_list</i> (a list of debugger commands) before executing any startup files or debugging the program. The sample option specification -do "property layout -notarget" illustrates a common use of this option (to inhibit the creation of a separate window for the target program). |
| -on <i>target_machine</i> | Debug the program or process on the specified target machine, where <i>target_machine</i> is a node name or node ID. |
| -target_type <i>target_type</i> | Specify the type of target machine; <i>target_type</i> must be "m68k" for SR10. |
| -input <i>pathname</i> | Read target program input from <i>pathname</i> . |
| -output <i>pathname</i> [-ao] | Direct target program output to <i>pathname</i> . With -ao, append output to <i>pathname</i> . |
| -errors <i>pathname</i> [-ae] | Direct target program error output to <i>pathname</i> . With -ae, append error output to <i>pathname</i> . To redirect error output and standard output to the same file, use the same pathname on both options or use "&1" as an argument to the -errors option. |
| <i>program_invocation</i> | Invoke <i>program_invocation</i> (the pathname of an executable image, plus any arguments) for debugging. This specification must be last on the dde command line. |
| -attach <i>process_id</i> | Attach to a running process identified by the UNIX pid <i>process_id</i> . Use the /bin/ps or /com/pst -un commands to get the pid of a process. |

NAME

delta — make a delta (change) to an SCCS file

SYNOPSIS

delta *-r*[SID] [*-s*] [*-n*] [*-glist*] [*-m*[*mrlist*]] [*-y*[*comment*]] [*-p*] *file*
...

DESCRIPTION

delta permanently introduces into the named SCCS file changes made to the file retrieved by **get**(1) (called the **g-file**, or generated file).

delta makes a delta to each named SCCS file. If you name a directory, **delta** behaves as though each file in the directory were specified as a named file, except that it silently ignores non-SCCS and unreadable files. If you specify a dash (-) in place of a filename, **delta** reads from the standard input, interpreting each line as the name of an SCCS file to be processed.

delta issues prompts on the standard output, depending on the presence of certain options in the SCCS file. See **admin**(1) for more information.

OPTIONS

The options described below apply independently to each named file.

- r*SID Uniquely identify which delta is to be made to the SCCS file. This option is necessary only if the same person (log-in name) has two or more outstanding **get** commands for editing (**get -e**) on the same SCCS file. The SID value specified with this keyletter is either the SID specified on the **get** command line or the SID to be made as reported by the **get**(1) command. If you specify an ambiguous SID or if you omit it when it must be present, a diagnostic results.
- s* Do not print the created delta's SID or the number of lines inserted, deleted, and unchanged in the SCCS file on standard output.
- n* Retain the edited **g-file** normally removed at completion of delta processing.
- glist* Specify a *list* of deltas to be ignored when the file is accessed at the change level (SID) created by this delta. See **get**(1) for the definition of *list*.
- m*[*mrlist*] If the SCCS file has the *v* flag set, you must supply a modification request (MR) number as the reason for creating the new delta. See **admin**(1) for more information.

If **-m** is not used, and the standard input is a terminal, **delta** issues the prompt "MRs?" on the standard output before it reads the standard input; if the standard input is not a terminal, **delta** does not issue a prompt. This particular prompt always precedes the prompt for comments (see the **-y** option). MRs in a list are separated by blanks and/or tab characters. An unescaped newline character terminates the MR list.

Note that if the **v** flag has a value, it is taken to be the name of a program (or shell procedure) that validates the correctness of the MR numbers. If a nonzero exit status is returned from the MR number validation program, **delta** assumes that the MR numbers were not all valid, and it terminates.

-y[comment] Specify arbitrary text to describe the reason for making the delta. A null string is considered a valid *comment*.

If you do not specify **-y**, and the standard input is a terminal, **delta** issues the prompt "comments?" on the standard output before it reads the standard input. If the standard input is not a terminal, **delta** does not issue a prompt. Use an unescaped newline character to terminate the *comment* text.

-p On the standard output, print the SCCS file differences before and after the delta is applied. Use a **diff(1)** format.

BUGS

Lines beginning with an SOH ASCII character (binary 001) cannot be placed in the SCCS file unless you escape the SOH. This character has special meaning to SCCS and causes an error if you do not escape it. Refer to **sccsfile(4)** for more information.

Avoid a **get** of many SCCS files, followed by a **delta** of those files, when the **get** generates a lot of data. Instead, use multiple **get/delta** sequences.

If you specify the standard input on the **delta** command line, you must use the **-m** and/or **-y** keyletters.

Comments are limited to text strings of 512 characters.

FILES

<i>g-file</i>	Existed before the execution of delta ; removed after completion of delta .
<i>p-file</i>	Existed before the execution of delta ; may exist after completion of delta .
<i>q-file</i>	Created during the execution of delta ; removed after completion of delta .
<i>x-file</i>	Created during the execution of delta ; renamed to SCCS file after completion of delta .
<i>z-file</i>	Created during the execution of delta ; removed during the execution of delta .

d-file Created during the execution of **delta**; removed after completion of **delta**.

/usr/bin/bdiff Program to compute differences between the “gotten” file and the *g-file*.

DIAGNOSTICS

Use **help(1)** for explanations.

SEE ALSO

admin(1), **bdiff(1)**, **cdc(1)**, **get(1)**, **help(1)**, **prs(1)**, **rmel(1)**, **sccs(1)**, **sccsfile(4)**

NAME

deroff – remove **nroff**, **troff**, **tbl**, and **eqn** constructs

SYNOPSIS

deroff [**-w**] *file* ...

DESCRIPTION

deroff reads each file in sequence and removes all **nroff**(1) and **troff**(1) command lines, backslash constructions, macro definitions, **eqn**(1) constructs (between **.EQ** and **.EN** lines or between delimiters), and table descriptions. It then writes the remainder on the standard output. **deroff** follows chains of included files (**.so** and **.nx** commands). If a file is already included, **deroff** ignores a **.so**, and a **.nx** terminates execution. If you do not specify an input file, **deroff** reads from the standard input file.

If you use the **-w** option, **deroff** outputs a word list, one word per line. All other characters are ignored. Otherwise, the output follows the original, with the deletions mentioned above. A word is a string that begins with a letter and contains letters, digits, and apostrophes. **deroff** removes trailing apostrophes from words.

BUGS

deroff is not a complete **troff**(1) interpreter, so it can be confused by subtle constructs. Most errors result in too much rather than too little output.

SEE ALSO

eqn(1), **tbl**(1), **troff**(1)

NAME

df – disk free

SYNOPSIS

df [**-laF** *nodefile*] [*filesystem* | *file* | *node_spec*] ...

DESCRIPTION

df prints the amount of free disk space available on the specified *filesystem* (for example, */dev/rp0a*), on the filesystem in which the specified *file* is contained, or on the specified node, *node_spec*.

If you don't specify a file system, **df** prints the free space for all normally mounted file systems. The reported numbers are in kilobytes.

The device filename is defined as follows:

<ctype><cnum + 16 + dnum><logvol>

Where:

ctype is the symbol for the device controller type and can be **wn** for winchester, **fl** for floppy, or **sm** for storage module.

cnum is a number between 0 and 3 inclusive, defining which controller is used with this device.

dnum is a number between 0 and 7 inclusive, defining the drive on which the controller resides.

logvol is a character between **a** and **o**, with **a** the first logical device and **o** the fifteenth.

OPTIONS

-l Report the number of inodes that are used and free. This option has no meaning on Domain/OS file systems, since there is no limit.

-a Print the values of total disk space, space available, space used, and percentage of disk space used.

If none of [*filesystem* | *file* | *node_spec*] are specified, **df** will report on status for all nodes in the network.

-F nodefile

nodefile contains information describing nodes for which **df** will print information. This file should contain node specifications (see *node_spec(5)*). There may only be one *node_spec* per line. A '#' in the first column of a line in the file causes that line to be treated as a comment.

WARNING

The device filename printed might not exist if the **mount** command is not used to mount the volume.

FILES

/etc/fstab List of normally mounted filesystems

SEE ALSO

fstab(5), *node_spec(5)*

NAME

diction, **explain** – print wordy sentences; thesaurus for **diction**

SYNOPSIS

diction [**-ml**] [**-mm**] [**-n**] [**-f pfile**] *file* ...
explain

DESCRIPTION

diction finds all sentences in a document that contain phrases from a database of bad or wordy diction. Each phrase is bracketed with []. Because **diction** runs **deroff(1)** before looking at the text, formatting header files should be included as part of the input. The default macro package is **-ms**.

explain is an interactive thesaurus for the phrases found by **diction**.

OPTIONS

The following options apply to **diction** only:

- mm** Specify **mm** as the macro package to be used instead of **ms**.
- ml** Make **deroff(1)** skip lists. Use this option if the document contains many lists of non-sentences.
- f pfile** Specify *pfile* as a pattern file to be used in addition to the default file.
- n** Suppress the default file.

BUGS

Using nonstandard formatting macros may cause incorrect sentence breaks. In particular, **diction** doesn't understand **-me**.

SEE ALSO

deroff(1), **ms(7)**, **mm(7)**

NAME

diff – differential file and directory comparator

SYNOPSIS

```
diff [ -l ] [ -r ] [ -s ] [ -Sname ] [ -cefn ] [ -biwt ] dir1 dir2
diff [ -cefn ] [ -biwt ] file1 file2
diff [ -Dstring ] [ -biw ] file1 file2
```

DESCRIPTION

If both arguments are directories, **diff** sorts the contents of the directories by name, and then runs the regular file **diff** algorithm (described below) on text files that are different. The **diff** command lists binary files that differ, common subdirectories, and files that appear in only one directory.

When run on regular files, and when comparing text files that differ during directory comparison, **diff** tells what lines must be changed in the files to bring them into agreement. Except in rare circumstances, **diff** finds the smallest sufficient set of file differences. If neither *file1* nor *file2* is a directory, then either may be given as **-**, in which case **diff** uses the standard input. If *file1* is a directory, **diff** uses a file in that directory whose filename is the same as the filename of *file2*, and vice versa.

There are several options for output format. The default output format contains lines of these forms:

```
n1 a n3,n4
n1,n2 d n3
n1,n2 c n3,n4
```

These lines resemble **ed**(1) commands to convert *file1* into *file2*. The numbers after the letters pertain to *file2*. In fact, by exchanging 'a' for 'd' and reading backward you can ascertain how to convert *file2* into *file1*. **diff**, like **ed**, abbreviates as a single number identical pairs where *n1* = *n2* or *n3* = *n4*.

Following each of these lines, **diff** prints all affected lines from the first file, flagged by **<**. Then **diff** prints all affected lines from the second file, flagged by **>**.

OPTIONS

The valid options for **diff** are presented below. The first four options (**-l**, **-r**, **-s**, and **-S**) pertain to comparisons between directories.

- l** Produce long output format. Pipe each text file **diff** through **pr**(1) to paginate it, and remember and summarize other differences after all text file differences are reported.
- r** Apply **diff** recursively to common subdirectories encountered.
- s** Report files that are the same, that are otherwise not mentioned.
- Sname** Start a directory comparison, beginning with file *name*.

The following options are mutually exclusive, except for **-b**, **-w**, **-l**, or **-t**, which you can specify with any of the others.

- e** Produce a script of **a**, **c**, and **d** commands for the editor **ed**, which will recreate *file2* from *file1*. In connection with **-e**, the following shell program may help maintain multiple versions of a file. Only an ancestral file (\$1) and a chain of version-to-version **ed** scripts (\$2,\$3,...) made by **diff** need be on hand. A latest version appears on the standard output.

```
(shift; cat $*; echo '1,$p') | ed - $1
```

diff adds extra commands to the output when comparing directories with **-e**, so that the result is a **sh**(1) script for converting text files that are common to the two directories, from their state in *dir1* to their state in *dir2*.
- f** Produce a script similar to that of **-e**, not useful with **ed**, and in the opposite order.
- n** Produce a script similar to that of **-e**, but in the opposite order and with a count of changed lines on each insert or delete command. This is the form used by **rcsdiff**.
- c[n]** Produce a comparison with lines of context. The default is to present three lines of context; you can change it. For example, **-c10** changes the lines of context presented to 10. With **-c** the output format is modified slightly. The output begins with identification of the files involved and their creation dates, then each change is separated by a line with a dozen *'s. The lines removed from *file1* are marked with -; those those added to *file2* are marked with +. Lines which are changed from one file to the other are marked in both files with !.

Changes that lie within <context> lines of each other are grouped together on output. (This is a change from the previous **diff -c** but the resulting output is usually much easier to interpret.)
- h** Do a fast, half-hearted job. This option is useful only when changed stretches are short and well separated, but it does work on files of unlimited length.
- Dstring** Create a merged version of *file1* and *file2* on the standard output, with C preprocessor controls included so that a compilation of the result without the defining *string* is equivalent to compiling *file1*, while defining *string* yields *file2*.
- b** Ignore trailing blanks (spaces and tabs), and cause other strings of blanks to compare equal.
- w** Ignore white space (blanks and tabs) completely. For example, "if (a == b)" will compare equal to "if(a==b)".

- i Ignore the case of letters. For example, "A" compares equal to "a".
- t Expand tabs in output lines. Normal or -c output adds character(s) to the front of each line which may change the indentation of the original source lines and make the output listing difficult to interpret. This option preserves the original source's indentation.

DIAGNOSTICS

Exit status is 0 for no differences; 1 for some; 2 for trouble.

BUGS

Editing scripts produced under the -e or -f option are naive about creating lines consisting of a single dot (.).

When you specify -b, -w, or -l, and **diff** is comparing directories, it first compares the files as in **cmp**(1), and then decides to run the **diff** algorithm if they are not equal. This may cause a small amount of spurious output if the files are identical except for insignificant blank-string or case differences.

FILES

/tmp/d????	
/usr/lib/diffh	for -h
/bin/diff	for directory diffs
/bin/pr	

SEE ALSO

cmp(1), **cc**(1), **comm**(1), **diff3**(1), **ed**(1), **pr**(1)
Using Your BSD Environment.

NAME

diff3 – three-way differential file comparison

SYNOPSIS

diff3 [**-exEX3**] *file1 file2 file3*

DESCRIPTION

diff3 compares three versions of a file, and publishes disagreeing ranges of text flagged with the following codes:

==== All three files differ.

====1 *file1* is different.

====2 *file2* is different.

====3 *file3* is different.

diff3 indicates the type of change that occurred in converting a given range of a given file to some other, in one of these ways:

f : *nl* **a** Text is to be appended after line number *nl* in file *f*, where *f* is 1, 2, or 3.

f : *nl* , *n2* **c** Text is to be changed in the range line *nl* to line *n2*. If *nl* = *n2*, the range may be abbreviated to *nl*.

The original contents of the range follows immediately after a **c** indication. When the contents of two files are identical, the contents of the lower-numbered file is suppressed.

OPTIONS

- e** Produce a script for the editor **ed**(1) that will incorporate into *file1* all changes between *file2* and *file3*; that is, changes that are normally flagged **====** and **====3**.
- E** Similar to the **-e** option. Produce a script for the text editor, but treat overlapping changes (that is, changes that are flagged with **====** in the normal listing) differently from the **-e** option. The edit script inserts the overlapping lines from both files, bracketed by "<<<<<<" and ">>>>>>" lines. **RCS merge** uses the **-E** option to ensure that overlapping changes in the merged files are preserved and brought to someone's attention.
- x (-3)** Produce a script to incorporate only changes flagged **====** (**====3**).
- X** Produce a script for the text editor **ed**, similar to the **-x** option, but treat overlapping changes (that is, changes that are flagged with **====** in the normal listing) differently. The edit script inserts the overlapping lines from both files, bracketed by "<<<<<<" and ">>>>>>" lines.

The following command applies the resulting script to **file1**.

```
(cat script; echo '1,$p') | ed - file1
```

EXAMPLE

For example, suppose lines seven and eight are changed in both **file1** and **file2**.

```
% diff3 -E file1 file2 file3 > edscript
% (cat edscript; echo '1,$p') | ed - file1
lines 1-6
of file1
<<<<<< file1
lines 7-8
of file1
=====
lines 7-8
of file3
>>>>>> file3
rest of file1
```

BUGS

Text lines that consist of a single dot (.) defeat **-e**.

FILES

```
/tmp/d3????
/usr/lib/diff3
```

SEE ALSO

comm(1), diff(1), ed(1)

NAME

dlty - delete a type

SYNOPSIS

dlty [*options*] *type_name*

DESCRIPTION

dlty deletes a type and any installed type manager.*type_name* (required) Specify the name of the type to be deleted.

OPTIONS

- n** *node_spec* Specify the node on which the type is to be deleted. You may also specify the entry directory of a volume mounted for software updates, as shown in the example below. If you omit the **-n** *node_spec* the type is deleted on the current node.
- l** List the type name/type identifier pair that is deleted.
- t**[*rait*] Delete a trait instead of a type.

EXAMPLES

```
$ dlty example_type -l
"example_type" 24BF9F41.100001FB deleted.
```

```
$ dlty example_type -n //test_vol -l
"example_type" 24BFA6F8.200001FB
deleted from volume //test_vol.
```

In the following example, the disk has been mounted for software updates. The disk's top level directory (cataloged as **/mount_disk** by the **/etc/mount(8)** command) must contain a "sys" directory. If it does not, you get a "types file not found" error.

```
$ mtvol w /mount_disk
$ dlty example_type -n /mount_disk -l
"example_type" 24BFB71E.200001FB deleted
from volume //my_node/mount_disk.
```

SEE ALSO

crtty(1), inty(1), lty(1), trty(1), mount(8)

NAME

dm commands - Display Manager Commands

DESCRIPTION

Following is a list of DM commands sorted by function.

CURSOR CONTROL COMMANDS:

al	Move cursor left 1 character position.
ar	Move cursor right 1 character position.
ad	Move cursor down 1 line.
au	Move cursor up one line.
as x y	Set scale factors for arrow keys, in raster units.
curs [-on -off]	Enable/disable cursor positioning via tn.
tl	Move cursor to the left edge of the pad.
tr	Move cursor to the end of the line.
tt	Move cursor to top edge of the window.
tb	Move cursor to the last line in the window.
twb {-l -r -t -b}	Move cursor to the specified window border.
th	Move cursor right to the next horizontal tab stop.
thl	Move cursor left to the next horizontal tab stop.
ts n1 n2...[-r]	Set tab stops in columns n1, n2, etc., optionally repeating the last interval.
tdm	Move cursor to the Display Manager's input window.
tlw	Move cursor to the previous window.
tn	Move cursor to the next window on the display.
tni	Move cursor to next unobscured icon on the display.
ti	Move cursor to the next window in which input is enabled.

PROCESS CREATION COMMANDS:

cp [-i -c <i>char</i>] <i>pathname</i> [-n <i>process_name</i>] [<i>args...</i>]	Create a new process, input and transcript pads, and associated windows; the process executes <i>pathname</i> ; -i makes the window an icon; -c specifies the icon character; -n names the process.
cpo <i>pathname</i> [-n <i>process_name</i>] [<i>args...</i>]	Create a process and execute <i>pathname</i> ; do not create pads or windows.
cps <i>pathname</i> [-n <i>process_name</i>] [<i>args...</i>]	Like cpo, except assign the process the SID 'user.server.none'.

PROCESS CONTROL COMMANDS:

- dq** [-s|-b|-c *nn*] Cause a quit fault, which normally terminates program execution; -s also stops the process; -b blasts the process, -c generates an arbitrary asynchronous fault with the specified hex status code.
- ds** Suspend execution of the process.
- dc** Continue execution of a suspended process.

WINDOW/PAD CREATION COMMANDS:

- ce** *pathname* Create an edit pad and associated window.
- cv** *pathname* Create a view, that is, a read only edit pad.
- cc** Create a copy of an existing window.

WINDOW CONTROL COMMANDS:

- wg** Grow or shrink a window.
- wge** Grow or shrink a window with feedback.
- wm** Move a window.
- wme** Move a window with feedback.
- wp** [*window_name*|*group_name*] [-t|-b]
Push (named) window (or window group) to bottom of pile if unobscured, else pop to top. -t and -b will force a window to the bottom or to the top.
- wc** [-f] [-q] [-a] [-s]
Close (delete) a window. Use -a to automatically close and delete a window after a ^Z and -s to reverse auto-close mode.
- wa** [-on|-off] Toggle auto-hold mode.
- ws** [-on|-off] Toggle window-at-a-time scroll mode.
- wh** [-on|-off] Toggle hold mode.
- wdf** [*n*] Set the *n*'th default window creation boundaries.

PAD CONTROL COMMANDS:

- pb** Move the bottom of the pad into window.
- pt** Move the top of the pad into window.
- pp** [-]*n* Move the pad forward [backward] *n* pages (*n* may be decimal fraction).
- pv** [-]*n* Move the pad forward [backward] *n* lines (*n* may be decimal fraction).
- ph** [-]*n* Move the pad *n* character positions horizontally (*n* may be decimal fraction).
- pn** *pathname* Save the pad under *pathname* (transcript pads only).

WINDOW GROUP AND ICON COMMANDS:

- icon** [*entry_name*] [-l] [-w] [-c 'char']
 Make a window or group of windows into an icon(s).
- wi** [*entry_name*] [-l] [-w]
 Make a window or group of windows invisible.
- wgra** *group_name* [*entry_name*]
 Add a window to a group of windows.
- wgrr** *group_name* [*entry_name*]
 Remove a window from a group of windows.
- cpb** *group_name* Display a list of the windows in a group.
- idf** Set icon positioning vector.

PAD EDITING COMMANDS:

Set modes:

- ro** [-on|-off] Change pad from write to read-only mode or vice versa.
- ei** [-on|-off] Change from insert to overstrike mode or vice versa.

Insert text:

- es** 'string' Insert 'string' at the current cursor position.
- en** Insert a new line character.
- er** *nn* Send raw hexadecimal data byte *nn* to user program.
- eef** Insert a stream end-of-file indicator.

Delete text:

- ed** Delete the character at the cursor position.
- ee** Delete the character immediately preceding the cursor.

Cut and Paste:

- xc** [-r] [-f *pathname* | *name*]
 Copy text into a paste buffer or file.
- xd** [-r] [-f *pathname* | *name*]
 Copy text into a paste buffer or file and delete text.
- xp** [-r] [-f *pathname* | *name*]
 Insert contents of paste buffer or file into pad.
- xi** [-f *pathname*] Copy display image to graphics map file for above cut and paste commands. Use -f to specify a file name.

Search:

- /regular exp/** Search forward in the pad for a string which matches the regular expression; for help on regular expressions, type help patterns.
- \regular exp** search backward in the pad for a string which matches the regular expression .
- // or ** repeat last search forward or backward.

sq	abort search.
sc [-on -off]	enable/disable case sensitivity for searches.
Substitute:	
s/re/replace/	Substitute the replacement text for all strings in the range which match the regular expression.
so/re/replace/	substitute the replacement text for the first string in each line in the range which matches the regular expression.
Miscellaneous:	
undo	undo file changes in an input pad or an edit pad; successive undos will undo further back in history.
pw	write edit pad to new file, but don't close pad or delete window.
echo [-r]	If a grow/move is in progress, then end feedback. Else begin text highlighting feedback if the cursor is on text.
abrt	Abort text or window feedback, abort a search, or clear mark stack.
case [-u -l -s]	Change the case of the letters in a marked text range. -s Switch to inverse case (default) -u Change to upper case -l Change to lower case

KEY DEFINITION COMMANDS:

kd key [[def] ke]	Set or display a DM key definition.
kbd n	Declare keyboard type; <i>n</i> must be '3' if your node has a Domain Programmable Keyboard (with numeric keypad); <i>n</i> must be '2' if your node has a Low Profile Keyboard; <i>n</i> must be ' ' if your node has an 880 keyboard; this command is only valid during node boot
&'prompt'	Write the optional prompt string in the input pad, then read a line of input

DISPLAY MANAGEMENT COMMANDS:**Login/Logout:**

l pers [group [org]]	Login (valid only when logged out); the 'l' is optional when preceded by the "login:" prompt.
lo [-f]	Logout (valid only when logged in)
ex	Exit DM to boot shell.
shut [-f]	Shutdown the system

Place/Clear Marks:

dr	Place a mark (for window control or cut and paste).
gm	Go to mark.

cms Clear mark stack.
rm Push last mark back on the stack.

Miscellaneous:

= Display line, column number, and x,y pixel values of current cursor position.

aa Acknowledge DM alarm.

ap Acknowledge DM alarm and pop the associated window .

bl [-l|-c] [*l_char*] [*r_char*]
 Check and/or balance delimiting characters.

env var [*value*] Set or display an environment variable; setting an environment variable is only valid during login startup.

bgc [-on|-off] Turn on or off the gray scale background color (monochrome monitor).

inv [-on|-off] Invert the screen to black on white, or vice-versa (monochrome monitor).

mono [-on|-off] Enable/disable monochrome mode (color monitor).

msg 'string' Display a message in the DM output window. rs Refresh the entire screen.

rw [-r] Refresh current window; -r option reenables window.

fl *pathname* [-i] Load a font to be used in later pads; -i indicates an icon font.

cmdf *pathname* Execute DM script.

NAME

dmwin – create a DM transcript/input pad pair and execute a program in it

SYNOPSIS

dmwin [*switches* | *program-name* [*program-arguments* ...]]

DESCRIPTION

The *dmwin* command creates a DM transcript/input pad pair and runs the specified program in it. It is intended to be used while you are running the Apollo shared X server, you are typing to an *xterm*(1), and you want to start a program that is going to try to "take over" the existing window pad as is common in the Apollo environment (Interleaf's -w option, /com/dspst, and many graphics programs behave this way). The tip-off that you should be using *dmwin* is receiving an error message similar to "attempted operation is illegal on this stream (stream manager/IOS)" when you try to start a program from an *xterm*(1).

dmwin exists to bridge a fundamental difference between the Apollo DM pad calls system of creating windows and the X window system (see "EXPLANATION" below).

You can also use *dmwin* from an *xterm* to run programs such as /com/mail and /com/dsee that need a DM pad for full-screen operation, rather than suffering them in line-by-line "teletype" mode in the *xterm* window.

If the first argument is present, it is either a switch or the name of the program to be run inside the wrapper. Any further arguments are passed as arguments to this program.

Permissible switches include the "-r" switch to specify rawmode, and the "-f filename" switch to specify the name of a file to be used for standard input.

IMPORTANT NOTE: Unless a "-f filename" switch is used, standard input is automatically assumed to be from "/dev/tty". This is done so that users may execute I. *dmwin* from a script with an "&" at the end of the command line to run it as a background process. It was found that when I. *dmwin* was executed in this manner, standard input was assumed to be from /dev/null, which caused problems when a call was executed in the newly created DM pad. Since it is expected that the priority of I. *dmwin* users will want to execute I. *dmwin* from an X window manager startup file as a background process, it seemed more appropriate to require the switch only when the I. *dmwin* user requires redirection of standard input. Hence, the "-f filename" switch.

The SHELL environment variable is used to identify the particular shell which is to be used. If this variable is not defined, the /com/sh shell is used.

If you use *dmwin* in a DM pad, it looks at the current standard input and output streams. If they are both DM pads already, it just invokes the given program with the given arguments. If either is not a DM pad, it creates a new DM transcript/input pad pair (using the DM's system for default size and position), and runs the given

program with the resulting streams as its standard ones. It uses the SHELL environment variable to determine what type of shell to run in the DM pad.

If you do not want to remember to type "dmwin" you can set up command scripts or aliases to make invocation of commonly-used Apollo commands look "Just Like Real Aegis" regardless of which type of window you are typing in.

Since Apollo graphics programs (pad calls included) cannot execute remotely over a network the way that X clients can, *dmwin* does not work if the xterm you are typing to is on a remote machine-- the graphics program will try to execute on the remote machine. The xterm must be on the local server.

EXAMPLE

```
prompt% dmwin ops -w &
```

EXPLANATION

In the Apollo environment, programs wishing to create and manipulate windows do so with pad calls. Practically all pad calls take an IOS (Input/Output Stream) stream ID as an input argument. This stream is assumed to be connected to an Apollo DM pad, and the pad call will fail if it is not. This stream is used as the channel over which to send a request to the DM to do something, such as to resize the window in which the pad appears. Furthermore, these streams are used in the initialization calls of the various Apollo graphics packages, if they are to work within an Apollo window.

Most Apollo native window/graphics programs "inherit" the DM transcript session which they ASSUME they were run from. This means that the programs do not try to create new windows and streams with pad calls, but instead use a standard stream (typically `ios_$stdout`) to operate on. The user's view in this case is that the transcript window is cleared of text, and the graphics session starts up in the same area. Examples of such programs are `/com/dspst` and `Interleaf` (if you give the '-w' option). These programs depend on the assumption that the standard output stream is in fact connected to a DM pad.

X does not have this stream/pad idea. In X, applications always create their own windows, which have window ID's, not stream ID's; they then can manipulate the window via Xlib calls; and when the program exits, its windows are deleted. Terminal sessions are not a special case as they are in the DM; instead, there is a particular kind of application called a terminal emulator, which acts very much like the Apollo VT100 emulator. The *xterm* program is the prime example of this. Such a program does X calls to create a window and draw text in it, and it also forks a shell whose standard streams are connected to pseudo-tty's controlled by it. So the shell and the programs it runs have standard streams connected to a pty, not a DM pad.

This fundamental difference in approach comes to a head when you are sitting in an xterm and you try to run a program that wants to inherit the xterm's non-existent DM pad stream IDs. Hence the error message: "attempted operation illegal on this stream (stream manager/IOS)."

dmwin is thus a "wrapper" function that, when run from an *xterm* creates a new DM transcript window, hooks up the standard streams to it, and then invokes the desired program.

Programs that do not make assumptions about their standard input/output stream (those that borrow the display or those that run themselves in a new window by using a `PAD_CREATE_WINDOW` call) will work when invoked from an *xterm(1)* without needing *dmwin*.

dmwin is written in C. The source is in `/domain_examples/X11/dmwin`

COPYRIGHT

Copyright 1988, 1989, Apollo Computer, Inc.
See *X(1)* for a full statement of rights and permissions.

AUTHOR

Dave Gorgen, Apollo Computer, Inc.

NAME

dspst – display process status graphically

SYNOPSIS

dspst [**-r** *n*] [**-p**] [**-l1**] [**-os**] [**-m**]
 [**-io**] [**-a**] [**-n** *node_spec*]
 [**-large** | **-small**]

DESCRIPTION

dspst displays process statistics in a graphical, bar-chart fashion within the current process window. The chart is updated periodically (see **-r** below). The default action of this command is to display the brief Domain/OS process list, all user processes and all I/O information in a font size automatically selected based on window size.

While **dspst** is running, the following keys are interpreted as follows:

All Keyboards:

CRTL/T	Move to top
CRTL/B	Move to bottom
RETURN	Exit
CRTL/N	Exit
CRTL/Y	Exit and save current image
Boxed up arrow	Scroll backward 1/2 window
Boxed down arrow	Scroll forward 1/2 window
Shifted up arrow	Scroll backward 1 line
Shifted down arrow	Scroll forward 1 line
EXIT or ABORT	Exit
SAVE	Exit and save current image

OPTIONS

-r <i>n</i>	Specify that the display should be repeatedly updated every <i>n</i> seconds. If this option is omitted, the display is updated every 4 seconds.
-p	Show process information.
-l1	Show Domain/OS and user-process information.
-os (default)	Show brief Domain/OS and full user-process information.
-m	Show missing CPU time.
-io (default)	Show I/O statistics.
-a	Show all information (same as -l1 -io -m).
-n <i>node_spec</i>	Specify remote node whose process statistics are to be listed.

- | | |
|-------------------------|--------------------------------------|
| -large (default) | Force use of large font for display. |
| -small | Force use of small font for display. |

EXAMPLES

1. Display Domain/OS, user process, and I/O status.

\$ **dspst**
2. Display Domain/OS, user process, and I/O status for the node named **//fred** using the large font.

\$ **dspst -n //fred -large**

NAME

dstdump – dump a COFF file's debugging information to stdout

SYNOPSIS

dstdump *input_pathname* [-m] [-b] [-s] [-l] [-u]

DESCRIPTION

The **dstdump** command dumps the debug symbol information from the COFF file *input_pathname* to the standard output device. This information includes the maker records in the **.mir** section, block records in **.blocks**, symbol records in **.symbols**, line number table in **.all** records, **organized by section and listed sequentially**. If you use any of the following command line options, however, **dstdump** displays information from the specified sections only:

- m **.mir** (maker information)
- b **.blocks** (compilation-unit and block information)
- s **.symbols** (constant, type and variable information)
- l **.lines** (line number tables)
- u **.unwind** (unwind records)

The **dstdump** command uses the command line parser, so it also accepts the standard command options listed in **HELP CL**.

The records that **dstdump** displays are defined in **/usr/include/mir.h** (mir records), **/usr/include/apollo/unwind.h** (unwind records), and **/usr/include/apollo/dst.h** (all others).

EXAMPLES

Display the module information from **y.bin**.

```
$ dstdump y.bin -m
```

```
-----
SECTION ".mir": 28 (0x1C) bytes
-----
```

```
Pascal compiler 68K=>PRISM Rev 8.6, Date: Tue Dec 13 21:26:40 1988
```

Display the block and symbol information from **y.bin**.

```
$ dstdump y.bin -b -s
```

```
-----
SECTION ".blocks": 150 (0x96) bytes
-----
```

0x000318> (size 26) type 1: COMP_UNIT

```

version      = (1, 3)
flags        = { pc_based_locs }
lang_type    = Pascal
number_of_blocks = 1
root_block_offset = 0x0022 -> 0x33a
section_table = 0x0054 -> 0x36c
file_table   = 0x0088 -> 0x3a0
data_size    = 0x0094 -> 0x3ac

```

0x000332> (size 8) type 28: GLOBAL_NAME_TAB

```

total length = 2
"x"

```

0x00033a> (size 40) type 4: BLOCK

```

{ has_aux_recs }
block_type    = program
flags         = { }
sibling_block = 0 (nil)
child_block   = 0 (nil)
noffset       = -0x0002 -> 0x338 "x"
symbols_start = [3, 0x0000000e] -> 0x00000296
n_of_code_ranges = 1
code_ranges:
  code_size    = 136 (0x88)
  code_start   = [4, 0x00000000]
  lines_start  = [1, 0x00000000] -> 0x00000270

```

0x000362> (size 10) type 51: AUX_SRC_RANGE

```

first_line = (file 1, line 1)
last_line  = (file 1, line 17)

```

0x00036c> (size 24) type 2: SECTION_TAB

index	base	address	section
1	100058	5:	.lines
2	80230	2:	.unwind
3	100070	6:	.symbols
4	801a8	1:	.text
5	100020	3:	.data

0x000384> (size 28) type 27: STRING_TAB

```

total length = 21
"//anode/my_dir/x.pas"

```

0x0003a0> (size 12) type 3: FILE_TAB

```

number_of_files = 1
files (dum name_offset):

```

1988/12/14 19:22:38-0x0016 -> 0x38a "//anode/my_dir/x.pas"

0x0003ac> (size 2) type 0: PAD

SECTION ".symbols": 140 (0x8c) bytes

0x000288> (size 14) type 27: STRING_TAB

total length = 8

"d"

"e"

"f"

"x"

0x000296> (size 20) type 46: VARIABLE

noffset = -0x0004 -> 0x292 "f"

loffset = 0x0064 -> 0x2ec [sect(6) - 28]

src_loc = (file 1, line 7)

type_desc = real32

attributes = { }

0x0002aa> (size 20) type 46: VARIABLE

noffset = -0x001a -> 0x290 "e"

loffset = 0x0055 -> 0x2f1

location in (0154, 02a8): .12

src_loc = (file 1, line 6)

type_desc = int32

attributes = { }

0x0002be> (size 20) type 46: VARIABLE

noffset = -0x0030 -> 0x28e "d"

loffset = 0x0048 -> 0x2f8

location in (0120, 0240): .0

src_loc = (file 1, line 6)

type_desc = int32

attributes = { }

0x0002d2> (size 20) type 48: ENTRY

noffset = -0x003e -> 0x294 "x"

loffset = 0x003a -> 0x2fe sect(6) - 24

src_loc = (file 1, line 1)

sig_desc = 0 (nil)

entry_number = 0

0x0002e6> (size 30) type 45: LOCPOOL

total length = 24

2ec: 25 69 1c 0e
2f1: 80 28 88 44 6d 0c
2f8: 80 28 88 0c 10
2fe: 25 69 18

NAME

du – summarize disk usage

SYNOPSIS

du [**-a**] [**-s**] [*names*]

DESCRIPTION

du gives the number of kilobytes contained in all files and, recursively, directories within each directory specified by *names*. If you don't specify a *names* argument, **du** prints information about the current directory, "." by default. A file that has two links to it is only counted once.

OPTIONS

- a** Generate an entry for each file.
- s** Print only the grand total of kilobytes for each of the specified *names*.

Absence of either the **-a** option or the **-s** option causes an entry to be generated for each directory only.

BUGS

If you don't use the **-a** option, **du** does not list nondirectories given as arguments.

If too many distinct linked files exist, **du** counts the excess files more than once.

SEE ALSO

df(1)

NAME

echo — echo arguments

SYNOPSIS

echo [**-n**] [*arg* ...]

DESCRIPTION

echo writes its arguments separated by blanks, and terminated by a newline, on the standard output. If you specify the **-n** option, **echo** doesn't add a newline to the 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 in a **/bin/sh** shell script, specify the following:

echo ... 1>&2

SEE ALSO

csh(1), **sh**(1)

NAME

ed — text editor

SYNOPSIS

ed [-] [*file*]

DESCRIPTION

ed is the standard text editor. **ed** operates on a copy of any file it is editing. Changes made in the copy have no effect on the file until you give a **w** (write) command. 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 **.** alone at the beginning of a line.

OPTIONS

- | | |
|-------------|--|
| <i>file</i> | If you specify a <i>file</i> argument, ed simulates an e command (see below) on the named file; that is, it reads the file into the buffer so that it can edit the file. |
| - | Suppresses the printing of explanatory output. You should use this option when the standard input is an editor script. |

REGULAR EXPRESSIONS

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. In the following specification for regular expressions the word *character* means any character except newline.

1. Any character except a special character matches itself. Special characters are the regular expression delimiter plus **[** **.** and sometimes **^** ***** **\$**.
2. **A .** matches any character.
3. **A ** followed by any character except a digit or **()** matches that character.
4. A nonempty string *s* bracketed **[s]** (or **[^s]**) matches any character in (or not in) *s*. In *s*, **** has no special meaning, and **]** may only appear as the first letter. A substring *a-b*, with *a* and *b* in ascending ASCII order, stands for the inclusive range of ASCII characters.
5. A regular expression of form 1-4 followed by ***** matches a sequence of 0 or more matches of the regular expression.
6. A regular expression, *x*, of form 1-8, bracketed **\(x \)** matches what *x* matches.
7. **A ** followed by a digit *n* matches a copy of the string that the bracketed regular expression beginning with the *n*th **\(** matched.

8. A regular expression of form 1-8, *x*, followed by a regular expression of form 1-7, *y* matches a match for *x* followed by a match for *y*, with the *x* match being as long as possible while still permitting a *y* match.
9. A regular expression of form 1-8 preceded by ^ (or followed by \$), is constrained to matches that begin at the left (or end at the right) end of a line.
10. A regular expression of form 1-9 picks out the longest among the leftmost matches in a line.
11. An empty regular expression stands for a copy of the last regular expression encountered.

Regular expressions are used in addresses to specify lines and in one command (see *s* below) to specify a portion of a line which is to be replaced. If it is desired to use one of the regular expression metacharacters as an ordinary character, that character may be preceded by \. This also applies to the character bounding the regular expression (often /) and to \ itself.

ADDRESSES

To understand addressing in *ed* it is necessary to know that at any time there is a *current line*. Generally speaking, the current line is the last line affected by a command; however, 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.
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; for example, -5 is understood to mean .-5.
9. If an address ends with + or -, then 1 is added (resp. subtracted). 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. To maintain compatibility with earlier versions of the editor, the character `^` in addresses is equivalent to `-`.

Commands may require zero, one, or two addresses. Commands that require no addresses regard the presence of an address as an error. Commands that 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. The special form `%` is an abbreviation for the `1,$` address pair.

COMMANDS

In the following list of `ed` commands, the default addresses are shown in parentheses. The parentheses are not part of the address.

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. Commands may also be suffixed by `n`, meaning the output of the command is to be line numbered. These suffixes may be combined in any order.

(.)a
<text>

The **append** command reads the given text and appends it to the addressed line. `"."` is left on the last line input, if there is any; otherwise it is left at the addressed line. Address 0 is legal for this command; *text* is placed at the beginning of the buffer.

(.,.)c
<text>

The **change** command deletes the addressed lines, then accepts input text that replaces these lines. `"."` is left at the last line input; if there is none, it is left at the line preceding the deleted lines.

(.,.)d

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

e *filename*

The **edit** command causes the entire contents of the buffer to be deleted, and the named file to be read in. `"."` is set to the last line of the buffer. The number of characters read is typed. *filename* is remembered for possible use as a default filename in a subsequent `r` or `w` command. If *filename* is missing, the remembered name is used.

E *filename*

This command is the same as **e**, except that no diagnostic results when no **w** has been given since the last buffer alteration.

f *filename*

The **filename** command prints the currently remembered filename. If you specify *filename*, the currently remembered filename is changed to *filename*.

(1,\$)g/*regular expression/command list*

The **global** command first marks every line that matches the given regular expression. For every such line, the given command list is executed with “.” initially set to that line. A single command or the first of multiple commands appears on the same line with the global command. All lines of a multiline list, except the last line, must be ended with \. **A**, **i**, and **c** commands and associated input are permitted; the “.” terminating input mode may be omitted if it would be on the last line of the command list. The commands **g** and **v** are not permitted in the command list.

(.)i

<text>

.

The **insert** command inserts the given text before the addressed line; “.” is left at the last line input, or, if there is none, at the line before the addressed line. This command differs from the **a** command only in the placement of the text.

(.,.+1)j

The **join** command joins the addressed lines into a single line; intermediate newlines simply disappear. “.” is left at the resulting line.

(.)kx

The **mark** command marks the addressed line with name *x*, which must be a lowercase letter. The address form ‘*x*’ then addresses this line.

(.,.)l

The **list** command prints the addressed lines in an unambiguous way. Non-graphic characters are printed in two-digit octal, and long lines are folded. The **l** command may be placed on the same line after any non-I/O command.

(.,.)ma

The **move** command repositions the addressed lines after the line addressed by *a*. The last of the moved lines becomes the current line.

(.,.)p

The **print** command prints the addressed lines. “.” is left at the last line printed. The **p** command may be placed on the same line after any non-I/O command.

(.,.)P

This command is a synonym for **p**.

- q** The **quit** command causes **ed** to exit, without doing an automatic write of a file.
- Q** This command is the same as **q**, except that no diagnostic results when no **w** has been given since the last buffer alteration.

(\$) **r** *filename*

The **read** command reads in the given file after the addressed line. If you did not specify a filename, **ed** uses the remembered filename, if any (see **e** and **f** commands). **ed** remembers the filename if there was no remembered filename already. Address 0 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. "." is left at the last line read in from the file.

- (...)**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. On each line in which a match is found, all matched strings are replaced by the replacement specified, 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 punctuation character may be used instead of / to delimit the regular expression and the replacement. "." is left at the last line substituted.

An ampersand (&) appearing in the replacement is replaced by the string matching the regular expression. The special meaning of & in this context may be suppressed by preceding it by \. The characters '\n' where *n* is a digit, are replaced by the text matched by the *n*th regular subexpression enclosed between \(and \) characters. When nested, parenthesized subexpressions are present, *n* is determined by counting occurrences of \(starting from the left.

Lines can be split by substituting newline characters into them. The newline in the replacement string must be escaped by preceding it by \.

One or two trailing delimiters may be omitted, implying the **p** suffix. The special form **s** followed by *no* delimiters repeats the most recent substitute command on the addressed lines. The **s** may be followed by the letters **r** (use the most recent regular expression for the left hand side, instead of the most recent left hand side of a substitute command), **p** (complement the setting of the *p* suffix from the previous substitution), or **g** (complement the setting of the *g* suffix). These letters may be combined in any order.

(...)**t***a*

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). "." is left on the last line of the copy.

(...)**u**

The **undo** command restores the buffer to its state before the most recent buffer-modifying command. The current line is also restored. Buffer-modifying commands are **a**, **c**, **d**, **g**, **i**, **k**, **m**, **r**, **s**, **t**, and **v**. For purposes of

undo, **g** and **v** are considered to be a single buffer-modifying command. **undo** is its own inverse.

When **ed** runs out of memory the full undo is not possible, and **u** can undo only the effect of the most recent **substitute** on the current line. This restricted **undo** also applies to editor scripts when you invoke **ed** with the **-** option.

(1, \$) v/regular expression/command list

This command is the same as the global command **g** except that the command list is executed **g** with **"."** initially set to every line except those matching the regular expression.

(1, \$) w filename

The **write** command writes the addressed lines onto the given file. If the file does not exist, it is created. The filename is remembered if there was no remembered filename already. If no file name is given, the remembered filename, if any, is used (see **e** and **f** commands). **"."** is unchanged. If the command is successful, the number of characters written is printed.

(1, \$) W filename

This command is the same as **w**, except that the addressed lines are appended to the file.

(1, \$) wq filename

This command is the same as **w** except that afterwards a **q** command is done, exiting the editor after the file is written.

(. +1) z or,

(. +1) zn

This command scrolls through the buffer starting at the addressed line; 22 (or **n**, if given) lines are printed. The last line printed becomes the current line. The value **n** is sticky, in that it becomes the default for future **z** commands.

(\$)= The line number of the addressed line is printed. **"."** is unchanged by this command.

!<shell command>

The remainder of the line after the **!** is sent to **sh(1)** to be interpreted as a command. The current line **"."** is unchanged.

(. +1, . +1) <newline>

An address alone on a line causes the addressed line to be printed. A blank line alone is equivalent to **. +1p** and is useful for stepping through text. If two addresses are present with no intervening semicolon, **ed** prints the range of lines. If they are separated by a semicolon, **ed** prints the second line.

If you send an interrupt signal **ed** prints the message **"?interrupted"** and returns to its command level.

NOTES

Some size limitations: 512 characters per line, 256 characters per global command list, 64 characters per file name, and, on mini computers, 128K characters in the temporary file. The limit on the number of lines depends on the amount of memory.

When reading a file, **ed** discards ASCII NUL characters and all characters after the last newline. It refuses to read files containing non-ASCII characters.

DIAGNOSTICS

“?name” for an inaccessible file; “?self-explanatory message” for other errors.

To protect against throwing away valuable work, a **q** or **e** command is considered to be in error, unless you have specified a **w** (write command) since the last buffer change. A second **q** or **e** is obeyed regardless.

BUGS

The **l** command mishandles interrupts.

The **undo** command causes marks to be lost on affected lines.

The **-x** option and the **x** command cannot be supported due to U.S. encryption protection policies.

FILES

/tmp/e*

edhup: Work is saved here if terminal hangs up

SEE ALSO

ex(1), **sed(1)**, *UNIX Text Processing*.

NAME

edfont – edit a character font

SYNOPSIS

edfont [*pathname* | *-v*]

DESCRIPTION

edfont is an interactive program with both menu-driven and command-line interfaces. It allows you to create, edit, and view character (glyph) font files. You can specify the font file with the **pathname** parameter, or use the "Open Font" entry in the "Font" menu. If the *-v* option is used, **edfont** will print its date of compilation and exit.

Generally, you must press the left mouse button <M1> to activate commands in the menu-driven interface. When you must enter a string (for example, when you designate which font you want to open) and there is a "Done" field on the menu, enter the string (plus <RETURN>), point to "Done" and press <M1> to activate. If "Done" does not appear when you must enter a string, simply type the string and press <RETURN> to activate the command.

Pressing the middle mouse button <M2> over the main graphics area toggles **edfont**'s mode between editing a glyph (displayed enlarged) and showing all of the glyphs at once. When all of the glyphs are displayed, pressing the left mouse button <M1> over a glyph will select it for editing. Also, when in this mode the menu to the right allows you to see the glyphs magnified by a factor of 1, 2, 3, or 4.

When using the menu-driven interface, you may notice that you cannot always select every menu choice. For example, you can't select "Open Font" if you already have one open, and likewise it's invalid to try to close a font when no font is open. When commands are invalid, as in these cases, their places on the menus are grayed out so that they can't be selected.

edfont lets you select a character through a variety of means. The utility interprets input in this manner: Any single character is taken as itself (e.g., **A** for **A**) Any character in single-quotes is taken as itself (e.g., ' ' for <space>) Any three-character string whose first character is a lowercase **c** has its final two characters interpreted as a compose sequence (e.g., **ca** for lowercase **a** with a circumflex accent) Any string of digits that begins with **0x** is interpreted as a hexadecimal code (e.g., **0x41** for uppercase **A**) Any string that begins with **0** (zero) is interpreted as octal (e.g., **0101** for **A**, but note that **0** alone is the character **0**) Any string that begins with a digit other than zero is considered to be decimal (e.g., **65** for **A**, but note that a single digit alone is interpreted as a character instead of a number) Any other string is considered to be an unknown name and is rejected.

For more information on compose sequences, see your system's User's Guide. For a list of decimal, octal, and hexadecimal values for the characters in Apollo's default character set, as well as a list of the compose sequences, see the files in the **/usr/pub** directory.

When you invoke **edfont**, it searches for two environment variables, **FONTSERVERS** and **FONTPATH**, which tell it where to look for the files it needs. If you set **FONTSERVERS** (using **setenv** from the shell) then **edfont** will look where you specify for the Patterns file and Config file it needs to operate. The Config file in turn tells **edfont** which font formats can be read and where to find the servers that understand those formats. If you do not set **FONTSERVERS** then **edfont** will look in the directory `/usr/apollo/lib/edfont`. If you do not set **FONTPATH** then **edfont**, when looking for a font file that you specify, will, if your name does not contain a path specifier, look first in the current directory and then in `/sys/dm/fonts`. This is equivalent to setting **FONTPATH** to `/sys/dm/fonts`. You can set **FONTPATH** to any other sequence of directories separated by colons (':').

When you invoke **edfont**, it sets default values for several variables. You can change those defaults using the appropriate command in the menu-driven interface or **set** in the command-driven interface. For more information on these interfaces see the section on command interfaces, below. The following table lists variables, their types, default values (if any), and purpose.

Variable/Type	Default	Description
fontpath /string	<code>/sys/dm/fonts</code>	List of directories, separated by colons, in which edfont should search for fonts
fontservers /string	<code>/usr/apollo/lib/edfont</code>	The search path for the font servers directory
fill /string	outline	The name of the current fill pattern
fontorigin /coord	none	The coordinate value that tells the number of pixels below and to the left of the font origin
fontsize /coord	none	The width and height of the font bounding box
fontspacing /coord	none	The horizontal and vertical font spacing (leading)
glyphoffset /coord	none	The offset of the current glyph from the font origin
glyphsize /coord	none	The width and height of the bitmap for the current glyph
glyphwidth /coord	none	The number of pixels from the right edge of the current glyph to the left edge of the next glyph
mask /string	src ^ dst	The current mask (raster operation)
edfont handles fonts created using Apollo's current and pre-SR10 formats, as well as Adobe BDF fonts.		

Menu Interface

Note: You can get additional information about any item on the display by pressing the right mouse button <M3> or by pressing the **HELP** key at the cursor position where you need help. This pops a help box. To return to the original display, move the cursor out of the help box.

Font

When you position the cursor here and press <M1>, **edfont** displays a menu with the following choices:

- Open Font
- Merge Font
- Font Params
- All Params
- Close Font
- Save As...
- Exit

Use these choices to open and close the font, merge in other fonts, save the font under another name, and examine or change the font's parameters. The **All Params** entry provides a powerful (and hence dangerous) means for simultaneously changing parameters of the font and thereby, all of the glyphs.

Glyph

If you press <M1>, you will see the following choices:

- Show ALL Glyphs
- Select Glyph
- Glyph Params Copy
- To Glyph
- Duplicate Glyph
- Delete Glyph
- Restore Glyph
- Rotate Glyph

Use these choices to perform all glyph operations. You can select a glyph, copy and duplicate glyphs, delete and restore, as well as examining all glyph parameters and rotating glyphs. The **Show ALL Glyphs** option duplicates the function of the middle mouse button, letting you see all glyphs at once.

Tools

If you press <M1>, you will see the following choices:

- Undo
- Run Commands
- Grid
- Metrics

Undo remembers your last changes (up to 10) performed on the current glyph. Undo does not work, however, on parameter changes.

Run Commands allows you to set up a file of commands and direct **edfont** to execute that file. For more information on the commands you can use, see the description of the command interface, below.

By default, both Grid and Metrics are turned on. If you turn off Grid, you no longer will see the pixel-by-pixel bitmap grid in the edit window. If you turn off Metrics, the glyph fills the edit window.

Metrics shows these three attributes of your glyph and font:

Origin and baseline (fine dotted line)
 Glyph-bounding box (long dashed line)
 Font-bounding box (short dashed line)

Draw When you position the cursor here and press <M1>, you will see the following choices:

Pixel Manipulate individual pixels

Freehand Draw freehand

Line Draw lines

Box Draw boxes

Circle Draw circles

Cut Select and delete a pixel or range of pixels

Copy Select and copy a pixel or range of pixels

Paste Paste in a pixel or range of pixels that you have previously cut or copied

Zoom Zoom in on a selected portion of the glyph

Note that after you Cut or Copy, **edfont** automatically changes the Draw mode to Paste, and after you Paste, **edfont** automatically changes the Draw mode back to what it was before. You can manually change the mode to something else (instead of Paste) after a Cut or Copy if you prefer. The cursor changes shape to alert you to the current mode.

Fill When you position the cursor here and press <M1>, you will see the following choices:

Outline this is the default

Solid (for filled rectangles and circles)

3/4-gray 75% gray

1/2-gray 50% gray

1/4-gray 25% gray

=stripes horizontal stripes

||stripes vertical stripes

//stripes right-leaning stripes

\stripes left-leaning stripes

squares fine checker board

You can modify this set of choices by changing the Patterns file used by **edfont**. This file is a BDF format font file, normally in /usr/apollo/lib/edfont, and can be edited by **edfont**. If you add new glyphs to this file, be sure to give them names for the menu.

edfont fills an entity, such as a circle or box, in a manner dependent upon the fill you choose. If you choose 50% gray, for

example, and then create a box, **edfont** turns on half of the pixels inside the box to create a 50% gray effect. If you choose 75% or 25% gray, **edfont** turns on proportionally more or fewer pixels to get the desired effect.

Mask

When you position the cursor here and press <M1>, you will see the following choices (where "src" means source, "dst" means destination (i.e., the current values in the region you select), and the other characters are logical operators):

Menu Choices

Logical Operation

set OFF	Assign zero (background) to selected region
set ON	Assign one (foreground) to selected region
set to src	Assign source values (from Fill mode) to selected region
dst XOR src	Assign source EXCLUSIVE OR destination to selected region (default)
dst OR src	Assign source OR destination to selected region
dst AND src	Assign source AND destination to selected region
dst BUT NOT src	Assign destination AND (complement of source) to new destination
set to NOT src	Assign complement of source to selected region
set to NOT dst	Assign complement of destination to selected region

The default mask **dst XOR src** toggles pixels. That is, if they already were black, they become white, and vice versa. However, if you are drawing in Freehand mode, this mask toggles the first pixel you cross and then sets the rest of the pixels you cross to that first pixel's value.

When you have a font open, the menu-driven interface also includes a row of boxes on the right side of the display labeled ***(lq{'**, ***(lq{<'**, **'>'** and **'}'**. These are for displaying the first, previous, next, and last glyph, respectively, in the current font. If the edit window is showing all the glyphs in a font, and if because of the size of the window not all of the glyphs can be viewed at once, then a similar set of keys can be used to scroll through the glyphs.

The name of the currently opened font is always shown in the upper right hand portion of the display, as is the currently selected glyph, current mask and fill pattern. The current glyph is also shown at real size in the bottom right hand portion of the display.

Command Interface

In addition to **edfont**'s menu-driven interface, you can use the following commands in the input pad at the bottom of the **edfont** window, or embed them in **edfont** scripts.

Commands (Arguments)	Description
<i>!shell-command</i>	Run a shell command in the edfont command window.
box <i>x1 y1 x2 y2</i>	Draw a box that is bounded by (<i>x1</i> , <i>y1</i>) and (<i>x2</i> , <i>y2</i>).
circle <i>x y r</i>	Draw a circle which has its center at (<i>x</i> , <i>y</i>) and a radius of <i>r</i> .
close [-save - nosave]	Close the font. If you specify -save , edfont saves your changes, while if you specify -nosave , edfont ignores them.
copy <i>glyphcode</i> [<i>fontfile</i>]	Copy the specified glyph to the current glyph. If you specify a <i>fontfile</i> , edfont copies the glyph from that font; otherwise, it copies the glyph from the current font.
delete	Delete the current glyph.
grid on off	Turn the bitmap grid on or off .
help [<i>command</i>]	Get a list of available commands, or get help on the specified command.
line <i>x1 y1 x2 y2</i>	Draw a line that begins at (<i>x1</i> , <i>y1</i>) and ends at (<i>x2</i> , <i>y2</i>).
metrics on off	Turn the font metrics display on or off .
next	Go to the next glyph in the current font.
open <i>fontfile</i>	Open the specified fontfile.
pixel <i>x y</i>	Draw a pixel at (<i>x</i> , <i>y</i>).
previous	Go to the previous glyph in the current font.
quit [-save - nosave]	Exit edfont , closing the current font (if one is open). See close for information on -save and -nosave .
rotate <i>degrees</i>	Rotate the current glyph by the specified number of degrees.
select <i>glyphcode</i>	Go to the specified glyph. For information on entering a glyph or <i>glyphcode</i> see the Description section above.
set <i>var=value</i>	Set <i>var</i> to the specified value. <i>var</i> can be one of the edfont 's parameters, as described in the Description section above.
source <i>filename</i>	Execute the command-script <i>filename</i> .
undo	Undo the last bitmap operation.
unzoom	Zoom out one level.
zoom <i>x1 y1 x2 y2</i>	Zoom in so that the view is filled with the box bounded by (<i>x1</i> , <i>y1</i>) and (<i>x2</i> , <i>y2</i>).

NAME

edmtdesc – edit magtape descriptor file

SYNOPSIS

edmtdesc *pathname* [*options*]

DESCRIPTION

edmtdesc allows you to create, list, and modify the magnetic tape descriptor object. The descriptor file provides information to the streams manager so that it can handle subsequent tape operations.

pathname (required) Specify name of magtape descriptor file to be created, listed, or edited.

OPTIONS

At least one of the following options must be specified.

- c** Create a new magtape descriptor object with the name given in the *pathname* argument.
- l [*var*...]** List the values of the variable(s) specified. If no variables are named, the entire magtape descriptor is listed.
- s [*var value*]...** Set the variable(s) indicated to the specified value(s). At least one variable/value pair is required if **-s** is specified. Multiple variable/value pairs are permitted, separated by blanks.

Variables

The variables known to **edmtdesc** are listed below, along with their types and default values. The variable types are: integer (int), Boolean (y/n), character string of *n* letters (c [*n*]), and date (in format yy/mm/dd.hh:mm).

Name	Type	Default	Definition
dev	c[1]	m	Device type ('m' for magtape, 'c' for cartridge)
u	int	0	Magtape unit number (normally 0)
lab	y/n	yes	'Yes' if magtape is ANSI labeled, 'no' if unlabeled
reo	y/n	no	'Yes' to reopen previously used volume, 'no' to open new volume ('yes' suppresses rewind)
clv	y/n	yes	'Yes' closes volume when file is closed, 'no' leaves volume open
spos	y/n	no	'Yes' saves volume position when volume is closed (for reopen), 'no' rewinds volume when closed

Name	Type	Default	Definition
vid	c[6]	-auto	Volume identifier (labeled volumes)
vacc	c[1]		Volume accessibility (labeled volumes)
own	c[14]	-auto	Volume owner (labeled volumes)
f	int*	1	file sequence number: integer or "cur" for current file, or "end" for new file at end of labeled volume
rf	c[1]	D	record format -- "F" for fixed length, "d" for variable length, "s" for spanned, "u" for undefined
bl	int	2048	block length, in bytes
rl	int	2048	(maximum) record length, in bytes
ascnl	y/n	yes	'Yes' for ASCII newline handling (strip newlines on write, supply them on read), 'no' for no newline handling
fsect	int	1	File section number (labeled volumes)
fid	c[17]		File identifier (labeled volumes)
fsid	c[6]		File set identifier (labeled volumes)
gen	int	1	Generation of file (labeled volumes)
genv	int	1	Generation version of file (labeled volumes)
cdate	date	-auto	Creation date of file (labeled volumes)
edate	date	-auto	Expiration date of file (labeled volumes)
facc	c[1]		File accessibility (labeled volumes)
sysc	c[xx]		System code (labeled volumes)
sysu	c[xx]		System use (labeled volumes)
boff	int	0	Buffer offset (labeled volumes, should be 0)

For cartridge tape (**dev c**), you must change the block length (**bl**) and the record length (**rl**) to be 512 or less and the record format to be fixed ("**rf F**").

EXAMPLES

Edit file **set_tape**; set the tape unit number to 1; declare tape as ANSI labeled.

```
$ edmtdesc set_tape -s u 1 lab yes
```

Create descriptor file **ct** for cartridge tape, blocking 4 records of maximum length 128 to each block.

```
$ edmtdesc ct -c -s dev c bl 512 rl 128 rf F
```


NAME

emt – emulate a dumb terminal

SYNOPSIS

emt [*pathname*]

DESCRIPTION

emt allows your node to emulate an ASCII terminal connected to another computer. This asynchronous connection exists through a stream opened on one of the node's SIO lines. **emt** also permits ASCII file transfer between your node and the remote host.

pathname (optional) Specify file containing **emt** commands.

Default if omitted: read commands from standard input

emt begins execution in local mode, and displays the following prompt:

emt>

To enter remote mode, press F1. (The **emt** command **dl** no longer exists.) In remote mode, your terminal operates as if it were physically connected to the remote computer ("host"). You can log on and enter remote host commands.

To return to local mode, press F1 again.

INPUT/OUTPUT STREAMS

emt uses the four standard streams: standard input, standard output, error input, and error output, as follows:

- **emt** commands are read from an **emt** command file or from standard input. The command filename may be specified on the command line or using the **emt run** command. Up to four levels of command files may be nested. When EOF is reached in a command file, commands are read from the previous file or from standard input. If EOF is reached on standard input, **emt** exits.
- Keystrokes to be sent to the host computer are read from standard input only.
- The **emt** command responses and all messages from the host are written to standard output.
- Error messages from Aegis system calls are written to error output. Optional monitoring (**monit**) may also be written to error output (or to a named file).

You may use redirection of standard input, command-line specification of a command file or the **emt run** command to automate **emt** usage and use **emt** in shell scripts. **emt** behaves slightly differently with regard to host transmissions, depending on which of these techniques you use and you may select the method that best suits your purpose.

When input is redirected to standard input ('**emt <emtfile1**'), lines in the command file that are sandwiched between F1 commands (enter/exit remote mode) are transmitted to the host. Other lines outside F1 commands are interpreted and executed as **emt** commands.

Contents of **emtfile1**:

Command	Description
interm lf	Sets input terminator.
outterm lf	Sets output terminator.
list	Lists emt state settings.
F1	Invokes remote mode (communication to host).
hello host	This and succeeding lines get sent to host.
goodbye host	Last line sent to host.
~li	emtesc char, specifies ' F1 ', return to local mode.
list	Back in local mode, lists emt state settings.
q	Exit from emt .

When a command file is invoked either via the command line (**emt emtfile2**) or by using the **run** command (**run emtfile2**), the behavior is different in that lines following **F1** commands are not transmitted to the host. This is because host transmissions are read from standard input and standard input has not been redirected to the file:

Contents of **emtfile2**:

Command	Description
interm lf	Sets input terminator.
outterm lf	Sets output terminator.
list	Lists emt state settings.
F1	Invokes remote mode (communication to host). All host input is now taken from the keyboard (or from standard input if it has been otherwise redirected). Finally user types ~l or presses F1 to return to local mode.
list	Local mode, emt commands read from emtfile2 again.
q	Exit from emt .

You may also use the **xmit** command to transmit a file (of commands or data) to the host. Use the **emt rcv** command to receive host transmissions to a Domain file.

TRANSFERRING FILES

You can transfer files using **emt**'s receive (**rcv**) or transmit (**xmit**) commands. **xmit** sends a Domain file to the remote host. **rcv** opens a Domain file to receive information from the remote host. For example, if you type (in local mode)

```
emt> xmit fileA
```

emt displays the following message:

```
Ready to transmit file fileA
```

Next, press **F1**. **emt** enters remote mode, and transmits fileA to the remote host.

If you type:

```
emt> rcv fileB
```

emt displays this message:

```
Ready to receive file fileB.
```

Next, enter remote mode by pressing **F1**. Use a remote host command to display the information that you want fileB to receive. **emt** automatically writes this and all subsequent host transmissions into fileB. To stop the **rcv**, press **F2**.

TRANSMISSION CONVENTIONS

Use the **emt** command **interm** to specify the line terminator used by the host. If you do not know what the host uses as a line terminator, experiment by changing **interm**. Use the **emt** command **outterm** to specify the line terminator to be transmitted to the host.

emt allows you to open only one Domain file at a time. If **emt** receives a **xmit** or **rcv** command while another Domain file is active, it closes the open Domain file, and executes the new command.

During remote mode, **emt** waits on both the keyboard and SIO line for characters to process, and monitors the data for characters of special interest to **emt**.

You can specify which keyboard characters **emt** should interpret by placing the keyboard in raw or cooked mode. In raw mode, **emt** passes all keyboard input (except the function keys, keys L1 through L12, and keys R1 through R4), directly to the host. Cooked mode lets you use many of the Display Manager's features for editing the input pad. **emt** places your keyboard in cooked mode by default.

COMMANDS

The following commands are available while running **emt**:

Command	Description
F1	Switch between local and remote modes.
F2	Interrupt a file transfer and close the file.
F3	Turn tee on or off. tee on causes emt to display file transmission records on the screen. You can use this feature to monitor file transfers, and decide if and when you should stop or interrupt a transfer. The default is tee on.
F8	Send a break to the host.
CTRL/F7	Display function key definitions.

These function keys may be simulated by typing the **emt** ESC character followed by the function key number (that is, ~1 for **F1**).

Command	Description
ae	Abort on error.
asconly notasconly	Sift out most non-printing ASCII codes. Eliminates triangles, allows BS, CR, ESC, FF, LF, TAB. The default is notasc .
break [n]	Set the break duration value to <i>n</i> milliseconds. The default is 200. If set to 0, the F8 (break) key does nothing.
close	Deactivate an rcv file. See the rcv command for related information.
code [xx none]	Set the host-command-code to the hexadecimal number <i>xx</i> . The default is none .
cooked	Place the keyboard in cooked mode. This enables many DM features for editing the input pad, and provides an escape sequence for sending control characters to the remote host. To send the host a CTRL character, precede the character with a tilde (~). The sequence ~_ transmits a delete character. To send the host a single tilde character, type ~. The emt default is cooked mode. Cooked mode always echos keystrokes, so it does not require a full duplex connection to the host. (See the raw command for related information.)

Note: The **cooked** and **raw** commands refer only to the transcript pad and keyboard input. The SIO line itself is always in **raw** mode.

emtesc [*chr*|*none*]

Set the **emt** escape character to *chr*. Use *none* to disable the escape character. Default is ~ for "cooked" mode, none for "raw" mode.

The following three commands are useful when standard input is redirected to a file of **emt** commands:

- f1** Enter remote mode (Simulate function key **F1**).
- f2** Terminate file transfer (Simulate function key **f2**).
- f3** Toggle tee mode (Simulate function key **F3**).
- hangup** Cause modem to break connection with the remote host.
- help** [*tctl*] Display information about **emt** commands or about **tctl** commands.

line {1|2|3|*pathname*}

Select the SIO line. Pathname must specify an SIO device descriptor (for example, */dev/sio2*). The default SIO line is 1 (*/dev/sio1*).

- l** Display the current SIO line, all **emt** switch settings and the receive filename, if any.

monit [*pathname*] Write every character received over the SIO line to *pathname*. If a filename is not specified, the previous specification or error output is used.

nomonit Stop monitoring.

quit End the **emt** session.

raw [**-echo**|-noecho] [**-lf**|-nolf]

Place the keyboard in "raw" mode. This sends keyboard input directly to the remote host, interpreting only function keys. The **-echo** option echos keystrokes on standard output; you should use it when the host is in half-duplex mode. The default is **-noecho**. The **-lf** option converts carriage return (CR) to line feed (LF) for lines echoed. The default is **-nolf**. (See the **cooked** command for related information.) Note: The **-echo** and **-lf** options are purely local functions that enable you to read what you type. They do not in any way change host/node transmissions.

rcv [-r] [-keys|-nokeys] [*pathname*]

Prepare the Domain file specified to receive remote host transmissions. If *pathname* already exists, **emt** appends the transmission to it, unless you specify **-r**. The receive begins when you enter remote mode **F1**. If you omit the *pathname*, **emt** uses the previous name, if any. The **-keys** option writes keystrokes to the file along with received data. The default is **-nokeys**.

emt allows you to interrupt an **rcv** command at any time by pressing **F2**. **emt** remains in whatever mode it was in, but keeps the **rcv** file active. When you are ready to continue receiving host transmissions, you may type **rcv** again (in local mode) without a filename, and **emt** uses the same **rcv** file.

If you omit filename and no **rcv** file is active, **emt** issues an error message. If you specify a new **rcv** file while another **rcv** file is active, **rcv** closes the active file, and prepares the new file to receive the transmission.

Use the **close** command to deactivate an **rcv** file.

tctl {*tctl commands*}

If you are running under Aegis, pass this command line to the shell command **tctl** to configure the SIO line. If this SIO line is not the default line, then you must use the **-line** command. The **speed** and **sync** commands have been superseded by this direct invocation of **tctl**. If only UNIX is installed, use **stty** to perform this action. If both UNIX and Aegis are installed, you can use either **tctl** or **stty**.

stty See **tctl**.

interm {*cr|lf|crlf|vax|'hex'*}

Select the input line terminator. The default is **crlf**.

outterm {*cr|lf|crlf|'hex'*}

Select the output line terminator. The default is **cr**. **emt** transmits the selected hexadecimal value as the terminator for each line.

xmit *pathname*

Prepare to transmit the Domain file specified to the remote host. If you omit *pathname*, or if you specify a file that does not exist, **emt** issues an error message. When you issue this command, **emt** remains in local mode. **emt** transmits the file when you press **F1**.

When **emt** completes the transfer, it closes the file and returns to the previous mode. **emt** does not send an end-of-file (EOF) signal to the remote host. If the host requires an EOF, enter remote mode and transmit it manually.

emt can also receive commands from the host. If the host transmits the sequence

```
host-command-code (emt command string) line-terminator
```

emt interprets the string as an **emt** command. Use the **emt** command **code** to define [host-command-code].

Line Terminators

emt Response

crlf	Converts sequence to a line feed, ignoring any null characters that may separate the pair.
cr	Converts sequence to a line feed and ignores LFs.
lf	Interprets it as a line feed, and ignores CRs.
vax	Interprets both CR and CR-LF as terminators and converts them to line feed.
'hex'	Converts the given hexadecimal value to LF.

NAME

eqn, neqn, checkeq – format mathematical text

SYNOPSIS

```
eqn [ -dxy ] [ -pn ] [ -sn ] [ -fn ] [ file ] ...
checkeq [ file ] ...
```

DESCRIPTION

eqn is a **troff**(1) preprocessor for typesetting mathematics on a phototypesetter or laser printer. **neqn** is used to format mathematical text on terminals. Usage is almost always as shown below:

```
eqn file ... | troff
neqn file ... | nroff
```

If you don't specify any files, **eqn** and **neqn** read from the standard input. A line beginning with **.EQ** marks the start of an equation; a line beginning with **.EN** marks the end of an equation. Neither line is altered, so you can define them in macro packages to get centering, numbering, and so on. You can also set two characters as delimiters; subsequent text between delimiters is treated as **eqn** input. The left and right delimiters can be identical. Use **delim off** to turn off delimiters. All text that is neither between delimiters nor between **.EQ** and **.EN** is passed through untouched.

The **checkeq** program reports missing or unbalanced delimiters and **.EQ/.EN** pairs.

Use spaces, tabs, newlines, braces, double quotes, tildes or circumflexes to separate tokens within **eqn**. Use braces {} for grouping. Generally speaking, anywhere a single character like *x* can appear, you can use a complicated construction enclosed in braces instead. A tilde (~) represents a full space in the output; a caret (^) half as much.

OPTIONS

- dxy** Set delimiters to characters *x* and *y*. Delimiters are more commonly set with **delim xy** between **.EQ** and **.EN**.
- pn** Change the point size of subscripts and superscripts to the value of *n*; normally they are reduced by three point sizes from the previous size.
- sn** Change the point size to the value of *n*.
- fn** Change the font to the value of *n*.

LANGUAGE

This section explains the **eqn** language.

Subscripts and superscripts are produced with the keywords **sub** and **sup**. Thus, **x sub 1** produces x_1 , the line **a sub i sup 2** produces a_i^2 and **e sup {x sup 2 + y sup 2}** produces $e^{x^2+y^2}$.

Fractions are made with **over**: **a over b** yields $\frac{a}{b}$.

The **sqrt** keyword makes square roots. **1 over sqrt {ax sup 2 +bx+c}** results in

$$\frac{1}{\sqrt{ax^2+bx+c}}$$

The keywords **from** and **to** introduce lower and upper limits on arbitrary things.

$$\lim_{n \rightarrow \infty} \sum_0^n x_i$$

is made with **lim from {n-> inf } sum from 0 to n x sub i**.

Left and right brackets, braces, and so on, of the right height are made with **left** and **right**. For example, **left [x sup 2 + y sup 2 over alpha right] = 1** produces

$$\left[x^2 + \frac{y^2}{\alpha} \right] = 1$$

The **right** clause is optional. Legal characters after **left** and **right** are braces, brackets, bars, **c** and **f** for ceiling and floor, and **''** for nothing at all (useful for a right-side-only bracket).

Vertical piles of things are made with **pile**, **lpile**, **cpile**, and **rpile**: **pile {a above b above c}** produces

$$\begin{array}{c} a \\ b \\ c \end{array}$$

There can be an arbitrary number of elements in a pile. **lpile** left-justifies, **pile** and **cpile** center, with different vertical spacing, and **rpile** right justifies.

Use **matrix** to make matrices; **matrix { lcol { x sub i above y sub 2 } ccol { 1 above 2 } }** produces

$$\begin{array}{cc} x_i & 1 \\ y_2 & 2 \end{array}$$

In addition, you can use **rcol** for a right-justified column.

Diacritical marks are made with **dot**, **dotdot**, **hat**, **tilde**, **bar**, **vec**, **dyad**, and **under**: **x dot = f(t) bar** produces

$$\dot{x} = \overline{f(t)}$$

y dotdot bar = n under produces $\ddot{y} = \underline{n}$ and **x vec = y dyad** produces $\vec{x} = \overrightarrow{y}$

You can change sizes and fonts with **size n** or **size ±n**, **roman**, **italic**, **bold**, and **font n**. You can change sizes and fonts globally in a document using **gsize n** and **gfont n**, or by using the command-line arguments **-sn** and **-fn**.

Normally, subscripts and superscripts are reduced by three point sizes from the previous size; this can be changed by the command-line argument **-pn**.

Successive display arguments can be lined up. Place **mark** before the desired line-up point in the first equation; place **lineup** at the place that is to line up vertically in subsequent equations.

Shorthands can be defined or existing keywords redefined with **define**; for example, **define thing % replacement %** defines a new token called *thing* that will be replaced by *replacement* whenever it appears thereafter. The % can be any character that does not occur in *replacement*.

Keywords like **sum** (Σ) **int** ($\int$) **inf** (∞) and shorthands like $\geq$ ($\geq$), $\rightarrow$ ($\rightarrow$), and $\neq$ ($\neq$) are recognized. Greek letters are spelled out in the desired case, as in **alpha** or **GAMMA**. Mathematical words like **sin**, **cos**, and **log**, are made roman automatically. You can use **troff(1)** four-character escapes like $\backslash dg$ (to produce the $\dagger$ character) anywhere. Strings enclosed in double quotes "..." are passed through untouched; this allows you to enter keywords as text, and you can use it to communicate with **troff** when all else fails.

BUGS

To make digits, parentheses, and so on, appear in boldface, you must quote them, as in **bold "12.3"**.

SEE ALSO

troff(1), **tbl(1)**, **ms(7)**, **me(7)**, **eqnchar(7)**

UNIX Text Processing

B. W. Kernighan and L. L. Cherry, *Typesetting Mathematics—User's Guide*

J. F. Ossanna, *NROFF/TROFF User's Manual*

NAME

error – analyze and disperse compiler error messages

SYNOPSIS

error [**-n**] [**-s**] [**-q**] [**-v**] [**-t** *suffixlist*] [**-I** *ignorefile*] [*name*]

DESCRIPTION

error analyzes and optionally disperses the diagnostic error messages, produced by a number of compilers and language processors, to the source file and line where the errors occurred. It can replace the painful, traditional methods of scribbling abbreviations of errors on paper, and permits you to view error messages and source code simultaneously without machinations of multiple windows in a screen editor.

error looks at the error messages, either from the specified file *name* or from the standard input, and attempts to determine which language processor produced each error message. It determines the source file and line number to which the error message refers, determines if the error message is to be ignored or not, and inserts the (possibly slightly modified) error message into the source file as a comment on the line preceding the line to which the message refers.

error does not insert error messages that can't be categorized by language processor or content into any file; instead, it sends them to the standard output. **error** touches source files only after all input has been read. If you specify the **-q** query option, **error** asks you to confirm any potentially dangerous (such as touching a file) or verbose action. Otherwise **error** proceeds on its merry business. If you specify the **-t** touch option and associated suffix list, **error** touches only those files with suffixes in the suffix list. You can also specify **-v** to ask **error** to invoke **vi(1)** on the files in which error messages were inserted; this means you don't have to remember the names of the files with errors.

error is intended to be run with its standard input connected via a pipe to the error-message source. Some language processors put error messages on their standard error file; others put their messages on the standard output. Hence, both error sources should be piped together into **error**. For example, when you use the **cs(1)** syntax,

```
make -s lint | & error -q -v,
```

error analyzes all the error messages produced by whatever programs **make** runs when making **lint**.

error knows about the error messages produced by: **make**, **cc**, **cpp**, **ld**, **lint**, and **f77**. **error** knows a standard format for error messages produced by the language processors, so it is sensitive to changes in these formats. Some error messages refer to more than one line in more than one file; **error** duplicates the error message and inserts it at all the places referenced.

error does one of the following six things with error messages.

Synchronize

Some language processors produce short errors describing which files they are processing. **error** uses these to determine the filename for languages that don't include the filename in each error message. **error** consumes these synchronization messages entirely.

- Discard** **error** discards messages from **lint** that refer to one of the two **lint** libraries, **/usr/lib/lib-lc** and **/usr/lib/lib-port**, to prevent accidentally touching these libraries. Again, **error** consumes these messages entirely. **error**.
- Nullify** **error** can nullify messages from **lint** if they refer to a specific function, that is known to generate uninteresting diagnostics. **error** does not insert nullified error messages into the source file, but writes them to the standard output. It takes the names of functions to ignore from either the file named **.errorrc** in your home directory, or the file named by the **-I** option. If the file does not exist, **error** does not nullify any error messages. If the file does exist, there must be one function name per line.
- Not file specific** **error** groups together messages that can't be intuited and writes them to the standard output before it touches any files. It does not insert them into any source file.
- File specific** **error** writes messages that refer to a specific file, but to no specific line, to the standard output when it touches that file.
- True errors** Error messages that can be intuited are candidates for insertion into the file they refer to.

Only true error messages are candidates for inserting into the file they refer to. **error** consumes other error messages entirely or writes them to the standard output. **error** inserts the error messages into the source file on the line preceding the line the language processor found in error. It turns each error message into a one line comment for the language, and flags it internally with the string **"###"** at the beginning of the error, and **"%%%"** at the end of the error. This makes pattern searching for errors with an editor easier, and allows you to remove the messages easily. In addition, each error message contains the source-line number for the line the message refers to. You can recompile a reasonably formatted source program with the error messages still in it, without having the error messages themselves cause future errors. For poorly formatted source programs in free-format languages, such as C or Pascal, you can insert a comment into another comment, which can wreak havoc on a future compilation. To avoid this, you should put language statements before comments in programs that have comments and source on the same line.

OPTIONS

- n** Do not touch any files; send all error messages to the standard output.
- q** Specifies that **error** queries you before it touches the file. You must answer **y** or **n** if you want **error** to continue. If you do not specify **-q**, you give **error** consent to touch all referenced files (except those referring to discarded error messages).
- v** After all files have been touched, overlay the visual editor **vi**, set up to edit all files touched, and positioned in the first touched file at the first error. If **vi** can't be found, try **ex** or **ed** from standard places.
- t** Take the following argument as a suffix list. **error** does not touch files whose suffixes do not appear in the suffix list. The suffix list is dot separated, and asterisk (*) wildcards work. Thus the suffix list
 ".c.y.foo*.h"
 allows **error** to touch files ending with **.c**, **.y**, **.foo*** and **.y**.
- s** Print statistics regarding the error categorization. This option is not too useful.

error catches interrupt and terminate signals, and if in the insertion phase, terminate what it is doing in an orderly fashion.

BUGS

Opens **/dev/tty** directly to do user querying.

Source files with links make a new copy of the file with only one link to it.

Changing a language processor's format of error messages may prevent **error** from understanding the error message.

Because **error** is purely mechanical, it does not filter out subsequent errors caused by 'floodgating' initiated by one syntactically trivial error. Humans are still much better at discarding these related errors.

error was designed for work on CRT's at reasonably high speed. It is less pleasant on slow-speed terminals, and has never been used on hardcopy terminals.

FILES

- ~/errorrc** Function names to ignore for **lint** error messages
- /dev/tty** User's teletype

NAME

esa – display address of external symbol

SYNOPSIS

esa *symbol_name*

DESCRIPTION

esa displays the address of an external symbol in an installed library. This command is primarily intended for system-level debugging.

symbol_name (required)

Specify the symbol whose address you wish to display. **esa** is case sensitive with respect to the symbol name. Lowercase must be used to refer to symbols defined in FORTRAN and Pascal programs. Mixed case may be used, as needed, for symbols defined in C programs.

EXAMPLES

This command displays the address of **gpr_\$init**. This symbol resides within the GPR library, which was installed at system start-up time.

```
$ esa gpr_$init
A1580C
$
```

NAME

ex, edit – text editor

SYNOPSIS

```
ex [ - ] [ -v ] [ -t tag ] [ -r ] [ +command ] [ -l ] name ...
edit [ options ]
```

DESCRIPTION

ex is the root of a family of editors, which also includes **edit** and **vi**(1). **ex** is a superset of **ed**(1), with the most notable extension being a display editing facility. Display-based editing is the focus of **vi**.

If you have not used **ed**, or you are a casual user, you will find that the editor **edit** is convenient for you. It avoids some of the complexities of **ex**, which is used mostly by systems programmers and persons very familiar with **ed**. **edit** uses the same options as **ex**.

If you want to use a display-based editor, see **vi**(1). The **vi** editor focuses on the display editing portion of **ex**.

See *UNIX Text Processing* for details concerning the use of these text editors.

OPTIONS

- Suppress all interactive user feedback. This is useful in processing editor scripts.
- v Invoke **vi**(1).
- t tag Edit the file containing the *tag* and position the editor at its definition.
- +command Begin editing by executing the specified editor search or positioning *command*.
- l Set lisp mode to indent appropriately for LISP code. Modify the parentheses, braces, and left and right double brackets commands, that is, (), {}, [[, and]], in **vi**(1) to have meaning for LISP.

BUGS

The **undo** command causes all marks to be lost on lines that are changed and then restored, if the marked lines are changed.

undo never clears the buffer modified condition.

The **z** command prints a number of logical rather than physical lines. More than a screenful of output may result if long lines are present.

File input/output errors don't print a name if you specify the command line option, –.

There is no easy way to do a single scan ignoring case.

The editor does not warn if text is placed in named buffers and not used before exiting the editor.

Null characters are discarded in input files, and cannot appear in resultant files.

FILES

/usr/lib/ex?.?strings	Error messages
/usr/lib/ex?.?recover	Recover command
/usr/lib/ex?.?preserve	Preserve command
/etc/termcap	Describes capabilities of terminals
~/exrc	Editor startup file
/tmp/Exnnnnnn	Editor temporary
/tmp/Rxnnnnnn	Named buffer temporary
/usr/preserve	Preservation directory

SEE ALSO

awk(1), ed(1), grep(1), sed(1), vi(1), termcap(5), environ(7);
UNIX Text Processing.

NAME

expand, **unexpand** – expand tabs to spaces, and vice versa

SYNOPSIS

```
expand [ -tabstop ] [ -tab1,tab2,...,tabn ] [ file ... ]  
unexpand [ -a ] [ file ... ]
```

DESCRIPTION

The **expand** command processes the named files or the standard input, writing the standard output with tabs changed into blanks. Backspace characters are preserved into the output and decrement the column count for tab calculations. **expand** is useful for pre-processing character files (before sorting, looking at specific columns, etc.) that contain tabs.

If you supply a single *tabstop* argument, the tabs are set *tabstop* spaces apart instead of the default, eight. If you specify multiple tabstops, the tabs are set at those specific columns.

unexpand puts tabs back into the data from the standard input or the named files, and writes the result on the standard output. By default, only leading blanks and tabs are reconverted to maximal strings of tabs. If you specify the *-a* option, tabs are inserted whenever they would compress the resultant file by replacing two or more characters.

NAME

expr – evaluate arguments as an expression

SYNOPSIS

expr *arg* ...

DESCRIPTION

The arguments are taken as an expression. After evaluation, **expr** writes the result on the standard output. Each token of the expression is a separate argument.

OPERATORS AND KEYWORDS

The operators and keywords are listed below. The list is in order of increasing precedence, with equal precedence operators grouped.

expr | *expr* Yields the first *expr* if it is neither null nor "0"; otherwise yields the second *expr*.

expr & *expr* Yields the first *expr* if neither *expr* is null or "0"; otherwise yields "0".

expr *relop* *expr* Where *relop* is one of < <= = != >= >, yields "1" if the indicated comparison is true; "0" if false. The comparison is numeric if both *exprs* are integers, otherwise it is lexicographic.

expr + *expr*

expr - *expr*

Addition or subtraction of the arguments.

expr * *expr*

expr / *expr*

expr % *expr*

Multiplication, division, or remainder of the arguments.

expr : *expr*

The matching operator compares the string first argument with the regular expression second argument; regular expression syntax is the same as that of **ed**(1). You can use the **\(...\)** pattern symbols to select a portion of the first argument. Otherwise, the matching operator yields the number of characters matched ("0" on failure).

(*expr*)

Parentheses for grouping.

EXAMPLES

To add 1 to the shell variable **a**:

a=`expr \$a + 1`

To find the filename part (least significant part) of the pathname stored in variable **a**, which may or may not contain **/**:

expr \$a : '.*^(.*)' ` ` \$a

Note the quoted shell metacharacters.

DIAGNOSTICS

expr returns the following exit codes:

- | | |
|---|---|
| 0 | if the expression is neither null nor 0 |
| 1 | if the expression is null or 0 |
| 2 | for invalid expressions |

SEE ALSO

csh(1), sh(1), test(1)

NAME

f77 – FORTRAN 77 compiler

SYNOPSIS

f77 [*option*] ... *file* ...

DESCRIPTION

f77 is the Domain/OS BSD interface to the preprocessors (**cpp**, **ratfor**, etc.), the Domain FTN compiler, and the link editor (**ld**). It accepts several types of arguments and interprets them as follows.

Arguments whose names end with **.f** are taken to be FORTRAN 77 source programs; **f77** compiles them, and leaves each object program in the current directory, in a file whose name is that of the source, with **.o** substituted for **.f**.

Arguments whose names end with **.F** are also taken to be FORTRAN 77 source programs. These are first processed by the C preprocessor before being compiled by **f77**.

Arguments whose names end with **.r** are taken to be Ratfor source programs; these are first transformed by the appropriate preprocessor, then compiled by **f77**.

Arguments whose names end with **.c** are taken to be C source programs and are compiled, producing a **.o** file.

Arguments whose names end with **.e** or **.s** (EFL or assembly source programs) are not supported by Domain/OS BSD.

The Domain/OS BSD **f77** command provides some unique options; not all standard UNIX options are available. **f77** recognizes the following as preprocessor options: **-C**, **-D**, **-I**, and **-Uname**. **f77** recognizes the following as link editor options: **-a**, **-l**, **-o**, **-r**, **-s**, **-t**, **-u**, **-x**, **-z**, **-L**, **-M**, and **-V**.

NOTE: You can typically improve run-time performance if you compile your code using the **-A cpu,ld** option with a processor-specific id. Use an argument appropriate to the CPU type of the machine(s) on which you will run the executable file. The option **-A cpu,3000**, for example, generates optimal code for the DSP90, DN330, DN560, DN570, DN580, DN3000, DN3500, DN4000, and DN4500 processors.

OPTIONS

The following options have the same meaning as in **cc(1)**. See **ld(1)** for load-time options.

- c** Suppress the linking phase of the compilation and force an object file to be produced, even if only one program is compiled.
- Dname=def** Define the *name* to the C preprocessor, as if by **#define**. If you do not supply a definition, define *name* as **1** (**.F** suffix files only).
- Dname** Define the *name* to the C preprocessor, as if by **#define**. If you do not supply a definition, define *name* as **1** (**.F** suffix files only).
- g** Produce additional debug information for **dbx** or **dde**.
- o output** Name the final output file *output* instead of **a.out**. If you specify a different name, the system leaves any existing **a.out** files undisturbed.

- O** Produce optimized code.
- p** Arrange for the compiler to produce code which counts the number of times each routine is called. If loading takes place, replace the standard startup routine by one which automatically calls **monitor(1)** at the start and arranges to write out a **mon.out** file at normal termination of execution of the object program. An execution profile can then be generated by use of **prof(1)**.
- pg** Causes the compiler to produce counting code in the manner of **-p**, but invokes a run-time recording mechanism that keeps more extensive statistics and produces a **gmon.out** file at normal termination. An execution profile can then be generated by use of **gprof(1)**.
- w** Suppress all warning messages.
- Wc,arg1[,arg2...]** Hand off the arguments *argi* to pass *c* where *c* is one of **p**, **0**, or **l**, indicating the preprocessor, compiler, or linker. You can use the **-W0** option to access **/com/ftn** options that are otherwise not available with the **/bin/f77** command. For example, **-W0,-l,foo** passes the option **-l foo** to the compiler. You can use the **-Wp** options to access **/bin/cpp** options that are otherwise not available with the **/bin/f77** command. For example, **-Wp,-C** passes the option **-C** to the C preprocessor. Note that no space is permitted between the comma and the option name.

The following **/com/ftn options** can be used with the **-W0** option. (default options are indicated by "(D)"):

- ac (D)** Produce absolute code. The alternative option is **-pic**, which forces the compiler to produce position-independent code.
- cond** Compile lines with **D** or **d** in column 1.
- ncond (D)** Ignore lines with **D** or **d** in column 1.
- config var1...varN** Set conditional compilation variables to true.
- dynm (D)** Allocate local variables on the stack (dynamic storage). Compare to **-save** option.
- exp** Generate expanded code listing (implies **-l**).
- nexp (D)** Suppress expanded code listing.
- ff** Activate free format: length of source lines may be up to 1024 characters;

	ampersand (&) in column 1 specifies a continuation line.
-frnd	Round and store floating-point numbers before comparisons. This option is implemented only on workstations with a 68881 processor.
-nfrnd (D)	Do not round and store floating-point numbers before comparisons. This option is implemented only on workstations with a 68881 processor.
-indexl	Disable some optimizations generated for subscript calculations, and cause all array indexing to use 4-byte integer arithmetic.
-nindexl (D)	Use all the optimizations generated for subscript calculations.
-info level	Control the output of informational messages. The five informational levels are 0, 1, 2 (default), 3, and 4.
-ninfo	Suppress informational messages.
-inlib <i>pathname</i>	Load <i>pathname</i> (a PIC binary file) at runtime and resolve global variable references. Thus you can use <i>pathname</i> as a library file for many different programs.
-inline <i>char</i>	Select <i>char</i> as an in-line comment designator. (Default <i>char</i> is "{".)
-l [<i>file</i>]	Generate compilation listing. If no <i>pathname</i> is specified, the listing is written to a file with the same name as the source file, but with the suffix .lst.
-nl (D)	Suppress compilation listing.
-l*1	Use logical*1 as the default logical type.
-l*2	Use logical*2 as the default logical type.
-l*4 (D)	Use logical*4 as the default logical type.
-msgs	Generate final error and warning count message.
-nmsgs	Suppress final error and warning count message.

-nclines	Suppress generation of COFF line number tables. Applies only to 680x0 target compilers.
-opt [n]	Perform global optimization. <i>n</i> is a single-digit integer in the range 0 to 4, indicating the level of optimization performed. -opt 0 is identical to the obsolete switch -nopt . If <i>n</i> is omitted, 3 is the default level. Increasing the optimization level results in faster execution of your program, at the expense of longer compilation time. See the <i>Domain FORTRAN Language Reference</i> manual for details.
-pic	Produce position-independent object code. The default is to produce absolute code.
-prasm (D)	Use Series 10000 assembly language format for the expanded listing generated by -exp . Note: this option applies only if used with the -exp option and with a compiler that generates Series 10000 code.
-nprasm	Use 68000 assembly language format for the expanded listing generated by -exp . Note: this option applies only if used with the -exp option and with a compiler that generates Series 10000 code.
-save	Allocate space for local variables in static storage, rather than on the stack. Compare with -dynm .
-type	Issue warning messages for variables not explicitly typed.
-ntype (D)	Suppress checking for explicit declarations of variables.
-xref	Insert a symbol map and cross reference in the listing file. This generates the listing file at <i>program_name.lst</i> , even if you omit -l .
-nxref (D)	Suppress creation of symbol map and cross-reference listing.

-xrs (D)	Save registers across a call to an external subprogram.
-nxrs	Do not assume that calls to external subprograms have saved the registers.
-zero	Initialize to zero all common blocks and statically allocated variables.
-nzero (D)	Do not initialize all common blocks and statically allocated variables.
The following <i>/bin/cpp</i> options can be used with the -Wp option:	
-C	Prevents the preprocessor from stripping C-style comments.
-Idir	Change the search path for #include files with names not beginning with a slash (/). Look first in the directory of the source file in which the #include directive occurs; then in directories named in this option; and finally, in directories on a standard list. (.F suffix files only.)
-Uname	Remove any initial definition of <i>name</i> .

The following options are peculiar to **f77**.

-C	Compile code to check that subscripts are within declared array bounds. For multidimensional arrays, check only the equivalent linear subscript.
-F	Apply the C preprocessor to .F files, apply the Ratfor preprocessors to .r files; put the result in the file with the suffix changed to .f, but do not compile.
-i2	Make the default integer constants and variables short (2 bytes). The default -i4 tells the compiler to make all integers four bytes.
-m	Apply the M4 macro preprocessor to each .r (Ratfor) source file before transforming it with the Ratfor preprocessor.
-q	Suppress printing of filenames and program-unit names during compilation.
-Rx	Use the string <i>x</i> as a Ratfor option in processing .r files.
-v	Print the version number of the compiler, and the name of each pass as it executes.

The following options are not supported in the Domain/OS BSD version of **f77**: **-d**, **-lg**, **-S**, **-onetrip**, **-1**, **-r8**, **-u**, **-Ex**, **-N**, and **-U**.

Other arguments are taken to be loader option arguments, or **f77**-compatible object programs, typically produced by an earlier run; or libraries of **f77**-compatible routines. These programs, together with the results of any compilations specified, are loaded (in the order given) to produce an executable program with the name **a.out**.

Domain/OS BSD EXTENSIONS

The **-A** option identifies Domain/OS BSD extensions to **f77** and **ld**; see **ld(1)** for additional information about this option.

-A nansi Pass to the C compiler a C source code file that does not comply with ANSI rules. Same as **/com/cc -nansi** option.

-A cpu,*id* Generate code for a particular class of processor; the default *id* is **any**. Values for *id* are as follows:

any Series 10000 code, if you're compiling on a
Series 10000 workstation, or 680x0 code, if
you're compiling on a 680x0-based workstation

a88k Series 10000 code

m68k 680x0-based code

160 DSP160 code

460 DSP460 code

660 660 code

90 DSP90 code

330 DN330 code

560 DN560 code

570 DN570 code

580 DN580 code

3000 DN3000, DN3500, DN4000, and DN4500 code

FPA1 Floating-Point Accelerator Board
for DN3500, DN4000, and DN4500

FPX Floating-Point Accelerator Board for DN5xx

PEB Performance Enhancement Board

The following values for *id* are synonyms: 160, 460, and 660. Also, the following values for *id* are synonyms: 90, 330, 560, 570, 580, and 3000. Using a processor-specific code generation mode can result in programs that run faster.

-A sys[*type*],*sys* Define the target system type (*sys*) for the compiled object; *sys* may be one of the following:

any	Version independent
bsd4.2	Berkeley version 4.2
bsd4.3	Berkeley version 4.3
sys5	UNIX System V
sys5.3	UNIX System V Release 3

-A run[type],sys Like **-A sys[type],sys** this option passes runtime information to the compiler and linker. Values for **sys** are the same as for **-A sys[type],sys**.

FILES

<i>file.[fFrc]</i>	Input file
<i>file.o</i>	Object file
<i>a.out</i>	Loaded output
<i>./fort[pid].?</i>	Temporary
<i>/usr/lib/cpp</i>	C preprocessor
<i>/usr/apollo/lib/ftn</i>	Compiler
<i>/usr/lib/libF77.a</i>	Intrinsic function library
<i>/usr/lib/libI77.a</i>	FORTRAN I/O library
<i>/usr/lib/libU77.a</i>	UNIX interface library
<i>mon.out</i>	File produced for analysis by prof(1)
<i>gmon.out</i>	File produced for analysis by gprof(1)

The following files are not supported.

<i>/usr/lib/f77pass1</i>	Compiler
<i>/lib/f1</i>	Pass 2
<i>/lib/c2</i>	Optional optimizer
<i>/usr/lib/libF77_p.a</i>	Profiling intrinsic function library
<i>/usr/lib/libI77_p.a</i>	Profiling FORTRAN I/O library
<i>/usr/lib/libU77_p.a</i>	Profiling UNIX interface library

DIAGNOSTICS

The diagnostics produced by **f77** itself are intended to be self-explanatory. The loader may produce occasional messages.

NOTES

After processing **.F** files, the C preprocessor leaves the resulting file in the current directory, in a file whose name is that of the source, with **.f** substituted for **.F**. If there is a previously existing **.f** file of the same name; it will be overwritten.

SEE ALSO

ratfor(1), **fpr(1)**, **fsplit(1)**, **ld(1)**, **ar(1)**, **dbx(1)**, **struct(1)**, **prof(1)**, **gprof(1)**, **cc(1)**, **intro(3f)**; the *Domain FORTRAN Language Reference*.

NAME

false, true – provide truth values

SYNOPSIS

true

false

DESCRIPTION

true and **false** are usually used in a Bourne shell script. They test for the appropriate status, “true” or “false” before running (or failing to run) a list of commands. The typical use of these commands is shown below.

```
while false  
do  
    command list  
done
```

DIAGNOSTICS

false has exit status nonzero.

SEE ALSO

csh(1), sh(1)

NAME

file – determine file type

SYNOPSIS

file [**-c**] [**-L**] [**-f** *ffile*] [**-m** *mfile*] *arg* ...

DESCRIPTION

The **file** command performs a series of tests on each argument in an attempt to classify it. If an argument appears to be ASCII, **file** examines the first 512 bytes and tries to guess its language. If an argument is an executable **a.out**, **file** prints the version stamp, if it is greater than zero.

The **file** command uses the file **/etc/magic** to identify files that have some sort of “magic number”, that is, any file containing a numeric or string constant that indicates its type. Commentary at the beginning of **/etc/magic** explains its format.

OPTIONS

- | | |
|------------------------|---|
| -c | Check the magic file for format errors. This validation is not normally carried out, for efficiency reasons. No file typing is done under -c . |
| -L | If the argument is a soft or symbolic link, test the file or directory the link references instead of the link itself. |
| -f <i>ffile</i> | Take the next argument to be a file containing the names of the files to be examined. |
| -m <i>mfile</i> | Use an alternate magic file, <i>mfile</i> . |

FILES

/etc/magic

NAME

find – find files

SYNOPSIS

find *pathname-list expression*
find *pattern*

DESCRIPTION

In the first form above, **find** recursively descends the directory hierarchy for each pathname in the *pathname-list* (that is, one or more pathnames) seeking files that match a Boolean *expression* written in the primaries given below.

The second form rapidly searches a database (*/usr/lib/find/find.codes*) for all pathnames that match *pattern*. The database is created by running the shell script */usr/lib/find/updatedb*, and is usually recomputed regularly by **cron**(8). The database contains the pathnames of all files that are publicly accessible. The */usr/lib/find/updatedb* script can be modified to create a list of site-specific local directories.

If escaped, normal shell “globbing” characters (*, ?, [, and]) may be used in *pattern*, but the matching differs in that no characters (for example, /) have to be matched explicitly. As a special case, a simple *pattern* containing no globbing characters is matched as though it were **pattern**; if any globbing character appears there are no implicit globbing characters.

EXPRESSIONS

In the following 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** *filename* True if the *filename* argument matches the current filename. Normal shell argument syntax may be used if escaped (watch out for '[', '?' and '*').
- perm** *onum* True if the file permission flags exactly match the octal number *onum* (see **chmod**(1)). If you prefix *onum* by a minus sign, more flag bits (017777, see **stat**(2)) become significant and the flags are compared: *(flags&onum)==onum*.
- type** *c* True if the type of the file is *c*, where *c* is **b**, **c**, **d**, **f**, **l**, or **s** for block special file, character special file, directory, plain file, symbolic link, or socket, respectively.
- links** *n* True if the file has *n* links.
- user** *uname* True if the file belongs to the user *uname* (log-in name or numeric user ID).
- nouser** True if the file belongs to a user not in */etc/passwd*.
- group** *gname* True if the file belongs to group *gname* (group name or numeric group ID).
- nogroup** True if the file belongs to a group not in */etc/group*.

-size <i>n</i>	True if the file is <i>n</i> blocks long (512 bytes per block).
-inum <i>n</i>	True if the file has inode number <i>n</i> .
-atime <i>n</i>	True if the file has been accessed in <i>n</i> days.
-mtime <i>n</i>	True if the file has been modified in <i>n</i> days.
-exec <i>command</i>	True if the executed command returns a zero value as exit status. You must punctuate the end of the command by an escaped semicolon. A command argument '{ }' is replaced by the current pathname.
-ok <i>command</i>	Like -exec but prints the generated command on the standard output, then reads the standard input and executes the command only if you respond by typing <i>y</i> .
-print	Always true; prints the current pathname.
-ls	Always true; causes the current pathname to be printed together with its associated statistics. These include (in the order they are printed) inode number, size in kilobytes (1024 bytes), protection mode, number of hard links, user, group, size in bytes, and modification time. If the file is a special file, the size field contains the major and minor device numbers instead. If the file is a symbolic link, the pathname of the linked-to file is printed preceded by "-->". The format is identical to that of ls -gilds . However, formatting is done internally, without executing the ls program).
-newer <i>file</i>	True if the current file was modified more recently than the argument <i>file</i> .
-cpio <i>file</i>	Write the current file on the argument <i>file</i> in cpio format.
-xdev	Always true; causes find not to traverse down into a file system different from the one on which current <i>argument</i> pathname resides.

OPERATORS

You can combine the primaries listed above, using the following operators (in order of decreasing precedence):

1. A parenthesized group of primaries and operators (parentheses are special to the shell and you must escape them).
2. The negation of a primary (! is the unary **not** operator).
3. Concatenation of primaries (the **and** operation is implied by the juxtaposition of two primaries).
4. Alternation of primaries (-o is the **or** operator).

EXAMPLES

To find all accessible files whose pathname contains **find**, enter

```
% find find
```

To remove all files named **a.out** or ***.o** that have not been accessed for a week, enter

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

FILES

/etc/passwd
/etc/group
/usr/lib/find/find.codes Coded pathnames database
/usr/lib/find/updatedb Shell script to create database

NOTES

If you're modifying anything in **/usr/lib/find**, you might be working in shared space (**/usr/lib/find** may be a link).

BUGS

The first form's syntax is painful, and the second form's exact semantics is confusing and can vary from site to site.

More than one **-newer** option does not work properly.

SEE ALSO

sh(1), **test(1)**
Relevant paper in February, 1983 issue of *login*.

NAME

finger – user information lookup program

SYNOPSIS

finger [*options*] *name* ...

DESCRIPTION

By default **finger** lists the log-in name, full name, terminal name and write status (as a "*" before the terminal name if write permission is denied), idle time, log-in time, and office location and phone number (if they are known) for each current user. (Idle time is minutes if it is a single integer, hours and minutes if a ":" is present, or days and hours if a "d" is present.)

A longer format also exists and is used by **finger** whenever you specify a list of people's names. Account names as well as users' first and last names are accepted. This format is multiline, and includes all the information described above as well as the user's home directory and log-in shell, any plan which the person has placed in the file **.plan** in his home directory, and the project he is working on from the file **.project**, also in the home directory.

finger can be used to look up users on a remote machine. Specify the user as "*user@host*". If you omit the username, **finger** provides the standard format listing on the remote machine.

OPTIONS

-m	Match arguments only on username.
-l	Force long output format.
-p	Suppress printing of the .plan files
-s	Force short output format.

FILES

/etc/utmp	who file
/etc/passwd	For users names, offices, ...
/usr/adm/lastlog	Last log-in times
~/.plan	Plans
~/.project	Projects

NOTES

finger performs poorly in large registries, unless you use the **-m** option.

BUGS

finger prints only the first line of the **.project** file.

The encoding of the **gcos** field is UCB dependent.

You cannot pass arguments to the remote machine, as **finger** uses an internet standard port.

A user information database is in the works and will radically alter the way the information that **finger** uses is stored. **finger** will require extensive modification when this is implemented.

Domain/OS does not support `/usr/adm/lastlog`.

SEE ALSO

`chfn(1)`, `who(1)`

NAME

fmt – simple text formatter

SYNOPSIS

fmt [*name* ...]

DESCRIPTION

fmt is a simple text formatter that reads the concatenation of input files (or standard input if you do not specify a file) and produces, on standard output, a version of its input with lines as close to 72 characters long as possible. **fmt** preserves spacing at the beginning of the input lines, blank lines, and interword spacing, in the output.

fmt is meant to format mail messages before you send them, but may also be useful for other simple tasks. For instance, within visual mode of the **ex** editor (for example, **vi**) the command

!}fmt

reformats a paragraph, making the lines even.

BUGS

The program was designed to be simple and fast; for more complex operations, the standard text processors are likely to be more appropriate.

SEE ALSO

nroff(1), **mail**(1)

NAME

fold – fold long lines for finite-width output device

SYNOPSIS

fold [*-width*] [*file ...*]

DESCRIPTION

fold is a filter that folds the contents of the specified files (or the standard input if you don't specify any files), breaking the lines to have a maximum width of *width*. The default *width* is 80 characters. *width* should be a multiple of eight if tabs are present, or the tabs should be expanded using **expand(1)** before executing **fold**.

BUGS

Executing **fold** on a file can interfere with any underlining.

SEE ALSO

expand(1), **fmt(1)**

NAME

fpr – print FORTRAN file

SYNOPSIS

fpr

DESCRIPTION

fpr is a filter that transforms files formatted according to FORTRAN's carriage-control conventions into files formatted according to UNIX line-printer conventions.

fpr copies its input onto its output, replacing the carriage-control characters with characters that produce the intended effects when printed using **lpr**(1). The first character of each line determines the vertical spacing as follows:

Character	Vertical Space Before Printing
Blank	One line
0	Two lines
1	To first line of next page
+	No advance

fpr treats a blank line as if its first character is a blank; deletes a blank that appears as a carriage-control character; changes a zero to a newline; changes a 1 to a form feed; and uses backspaces to simulate the effects of a "+".

EXAMPLES

```
% a.out | fpr | lpr
```

```
% fpr < f77.output | lpr
```

BUGS

Results are undefined for input lines longer than 170 characters.

NAME

from – who is my mail from?

SYNOPSIS

from [*-s sender*] [*user*]

DESCRIPTION

from prints the mail header lines in your mailbox file, to show who sent you mail. If *user* is specified, it uses the mail file for the login *user*. If the *-s* option is given, **from** prints only the header lines for mail sent by *sender*.

FILES

*/usr/spool/mail/**

SEE ALSO

biff(1), *mail(1)*

NAME

fsplit – split a multiroutine FORTRAN file into individual files

SYNOPSIS

fsplit [**-e** *efile*] ... [*file*]

DESCRIPTION

fsplit takes as input either a file or standard input containing FORTRAN source code. It attempts to split the input into separate routine files of the form *name.f*, where *name* is the name of the program unit (for example, function, subroutine, block data, or program). The name for unnamed block-data subprograms has the form **blkdta***NNN.f* where *NNN* is three digits, and a file of this name does not already exist. For unnamed main programs the name has the form **main***NNN.f*. If there is an error in classifying a program unit, or if *name.f* already exists, the program unit is put in a file of the form **zzz***NNN.f* where **zzz***NNN.f* does not already exist.

Normally each subprogram unit is split into a separate file. When you use the **-e** option, only the specified subprogram units are split into separate files.

EXAMPLE

The following command splits **readit** and **doit** into separate files.

fsplit -e readit -e doit prog.f

DIAGNOSTICS

If **fsplit** does not find the names you specify in the **-e** option, it writes a diagnostic to standard error.

BUGS

fsplit assumes the subprogram name is on the first non-comment line of the subprogram unit. Non-standard source formats may confuse **fsplit**.

It is hard to use **-e** for unnamed main programs and block data subprograms, because you must predict the created filename.

NAME

fst – print fault status information

SYNOPSIS

fst [**[-s]** **[-r]** | **[-a]**] **[-u** *n*]

DESCRIPTION

fst prints information about the most recent fault that occurred in the process. The information always includes the fault status, the program counter (PC) at which the fault occurred, and a textual description of the error as reported by the system call **error_\$(print)**. **fst** is intended for system-level debugging.

If you are using a Peripheral Bus Unit (PBU) device, you can get fault information by using the **-u** option (see below).

fst is obsolete and is valid only when running in INPROCESS compatibility mode with the **inprocess** variable set and all commands running in-process. Use the command **tb -full** instead of **fst**.

OPTIONS

- r** Print the contents of the CPU general registers when the fault occurred.
- s** Print the supervisor PC, entry control block (ECB), and status register (SR) if the fault occurred in supervisor mode. This option is ignored if the fault occurred in user mode.
- a** Print all available fault information. (Prints the same information as both **-s** and **-r**.)
- u** *n* Print the same information as both **-s** and **-r** for faults caused by the PBU interrupt handler for unit *n*.

EXAMPLES

\$ **fst -a**

```
Fault Diagnostic Information
Fault Status    = 00120010:
process quit (from OS / fault handler)
User Fault PC  = 000157FC
D0-D7: 00120010 00000000 00000002 FFFFFFFE 00000008 00000006 \
00000182 00000004
A0-A7: 0020A452 00E2F22E 0020A3D4 0020A450 00E2F174 0000C92C \
002746B4 002746AC
Supervisor ECB = 00000000
Supervisor SR  = 0000
Supervisor PC  = 00000000
```

NAME

ftp – ARPANET file transfer program

SYNOPSIS

ftp [-v] [-d] [-l] [-n] [-g] [host]

DESCRIPTION

ftp is the user interface to the ARPANET standard File Transfer Protocol (FTP). The program allows you to transfer files to and from a remote network site.

You can specify the client host with which **ftp** is to communicate on the command line. If you do, **ftp** immediately attempts to establish a connection to an FTP server on that host; otherwise, **ftp** enters its command interpreter and awaits instructions from you. When **ftp** is awaiting commands from you, it displays the prompt "ftp>".

COMMANDS

ftp recognizes the following commands:

! [*command* [*args*]]

Invoke an interactive shell on the local machine. If you specify arguments, **ftp** takes the first to be a command to execute directly, with the rest of the arguments as its arguments.

\$ *macro-name* [*args*]

Execute the macro *macro-name* that was defined with the **macdef** command. Arguments are passed to the macro unglobbed.

account [*passwd*]

Supply a supplemental password required by a remote system for access to resources once a login has been successfully completed. If you do not specify an argument, **ftp** prompts you for an account password in a non-echoing input mode.

append *local-file* [*remote-file*]

Append a local file to a file on the remote machine. If you do not specify *remote-file*, **ftp** uses the local filename, after applying the changes required by any **ntrans** or **nmap** setting, to name the remote file. **ftp** uses the current settings for **type**, **form**, **mode**, and **structure**.

ascii

Set the file transfer type to network ASCII. This is the default type.

bell

Arrange that a bell be sounded after each file transfer command is completed.

binary

Set the file transfer type to support binary image transfer.

bye

Terminate the FTP session with the remote server and exit **ftp**. An end-of-file also terminates the session and exits.

case

Toggle remote computer filename case-mapping during **mget** commands. When **case** is on (the default is off), remote computer filenames with all letters in uppercase are written in the local directory with the letters mapped to lowercase.

cd remote-directory

Change the working directory on the remote machine to *remote-directory*.

cdup

Change the remote-machine working directory to the parent of the current remote-machine working directory.

close

Terminate the FTP session with the remote server, and return to the command interpreter. Any defined macros are erased.

cr

Toggle carriage-return stripping during ASCII-type file retrieval. Records are denoted by a carriage-return/linefeed sequence during ASCII-type file transfer. When **cr** is on (the default), carriage returns are stripped from this sequence to conform with the UNIX single-linefeed record delimiter. Records on non-UNIX remote systems may contain single linefeeds; when an ASCII-type transfer is made, you can distinguish these linefeeds from a record delimiter only when **cr** is off.

delete remote-file Delete the file *remote-file* on the remote machine.

debug [debug-value]

Toggle debugging mode. If you specify an optional *debug-value*, **ftp** uses it to set the debugging level. When debugging is on, **ftp** prints each command sent to the remote machine, preceded by the string "-->".

dir [remote-directory] [local-file]

Print a listing of the directory contents in the directory, *remote-directory*, and, optionally, place the output in *local-file*. If you do not specify a directory, **ftp** uses the current working directory on the remote machine. If you do not specify a local file, or *local-file* is -, **ftp** sends output to the terminal.

disconnect

A synonym for **close**.

filetype type-name [local | remote | both]

Set the object type of Domain/OS or user-defined objects before transferring them with **get** or **put** operations. If you do not specify an argument, **filetype** displays the current settings for the file type used for transfers. To set a file type, specify the desired file type *type-name* (which can be any Domain/OS defined or user-defined type such as "coff," "unstruct," "uasc"), and specify whether the file type applies to **get** (*local*), **put** (*remote*), or both (*both*) operations. If you do not use **filetype** to set the file type, the files will be assigned the default type, "unstruct". If you attempt to specify *remote* and the remote system does not support **filetype**, the remote filetype remains unchanged. If you specify **filetype type-name** without any options, the default is *both*.

form format

Set the file transfer form to *format*. The default and only supported format is **file**.

get remote-file [*local-file*]

Retrieve the *remote-file* and store it on the local machine. If you do not specify the local filename, **ftp** gives it the same name it has on the remote machine, subject to alteration by the current **case**, **ntrans**, and **nmap** settings. **ftp** uses the current settings for **type**, **form**, **mode**, and **structure** while transferring the file.

glob

Toggle filename expansion for **mdelete**, **mget** and **mput**. If you turn globbing off with **glob**, **ftp** takes the filename arguments literally and does not expand them. Globbing for **mput** is done as in **cs(1)**. For **mdelete** and **mget**, each remote filename is expanded separately on the remote machine and the lists are not merged. Expansion of a directory name is likely to be different from expansion of an ordinary filename: the exact result depends on the foreign operating system and FTP server. You can preview the results by executing '**mls remote-files -**'. Note: **mget** and **mput** are not meant to transfer entire directory subtrees of files. You can do that by transferring a **tar(1)** archive of the subtree (in binary mode).

hash

Toggle hash-sign (#) printing for each data block transferred. The size of a data block is 1024 bytes.

help [*command*]

Print an informative message about the meaning of *command*. If you do not specify an argument, **ftp** prints a list of the known commands.

lcd [*directory*]

Change the working directory on the local machine. If you do not specify a *directory*, **ftp** uses your home directory.

ls [*remote-directory*] [*local-file*]

Print an abbreviated listing of the contents of a directory on the remote machine. If you do not specify *remote-directory*, **ftp** uses the current working directory. If you do not specify a local file, or if *local-file* is **-**, **ftp** sends the output to the terminal.

macdef *macro-name*

Define a macro. Subsequent lines are stored as the macro *macro-name*; a null line (consecutive newline characters in a file or carriage returns from the terminal) terminates macro input mode. There is a limit of 16 macros and 4096 total characters in all defined macros. Macros remain defined until you execute a **close** command. The macro processor interprets '\$' and '\' as special characters. A '\$' followed by a number (or numbers) is replaced by the corresponding argument on the macro-invocation command line. A '\$' followed by an 'i' signals that macro processor that the executing macro is to be looped. On the first pass '\$i' is replaced by the first argument on the macro-invocation command line, on the second pass it is replaced by the second argument, and so on. A '\' followed by any character is replaced

by that character. Use the ``` to prevent special treatment of the `'$'`.

mdelete [*remote-files*]

Delete the *remote-files* on the remote machine.

mdir *remote-files local-file*

This command works like **dir**, except that you can specify multiple remote files. If interactive prompting is on, **ftp** prompts you to verify that the last argument is indeed the target local file for receiving **mdir** output.

mget *remote-files*

Expand the *remote-files* on the remote machine and execute a **get** for each filename thus produced. See **glob** for details on the filename expansion. Resulting filenames are then processed according to **case**, **ntrans**, and **nmap** settings. Files are transferred into the local working directory, which you can change with '**lcd directory**'; You can create new local directories with '**mkdir directory**'.

mkdir *directory-name*

Make a directory on the remote machine.

mls *remote-files local-file*

This command is like **ls**, except that you can specify multiple remote files. If interactive prompting is on, **ftp** prompts you to verify that the last argument is indeed the target local file for receiving **mls** output.

mode [*mode-name*]

Set the file transfer **mode** to *mode-name*. The default and only supported *mode-name* is **stream**.

mput *local-files*

Expand wildcards in the list of local files given as arguments and execute a **put** for each file in the resulting list. See **glob** for details of filename expansion. Resulting filenames are then processed according to **ntrans** and **nmap** settings.

nmap [*inpattern outpattern*]

Set or unset the filename-mapping mechanism. If you do not specify an argument, the filename-mapping mechanism is unset. If you specify an argument, **nmap** maps remote filenames during **mput** commands and **put** commands issued without a specified remote-target filename. and maps local filenames during **mget** commands and **get** commands issued without a specified local-target filename. This command is useful when you are connecting to a non-UNIX remote computer with different file-naming conventions or practices.

The mapping follows the pattern set by *inpattern* and *outpattern*. *Inpattern* is a template for incoming filenames (which may have already been processed according to the **ntrans** and **case**

settings). Include the sequences '\$1', '\$2', ..., '\$9' in *inpattern*, if you want variable templating. Use '\ ' to prevent this special treatment of the '\$' character. **nmap** treats all other characters literally, and uses them to determine the **nmap** *inpattern* variable values. For example, given *inpattern* \$1.\$2 and the remote filename "mydata.data", \$1 has the value "mydata", and \$2 has the value "data".

The *outpattern* determines the resulting mapped filename. The sequences '\$1', '\$2', ..., '\$9' are replaced by any value resulting from the *inpattern* template. The sequence '\$0' is replaced by the original filename. Additionally, the sequence '[seq1,seq2]' is replaced by *seq1* if *seq1* is not a null string; otherwise it is replaced by *seq2*. For example, the command **nmap \$1.\$2.\$3 [\$1,\$2].[\$2,file]** yields the output filename **myfile.data** for input filenames **myfile.data** and **myfile.data.old**, **myfile.file** for the input filename **myfile**; and **myfile.myfile** for the input filename **.myfile**. You can include spaces in *outpattern*, as in the example: **nmap \$1 [sed "s/ *\$//" > \$1** . Use the '\ ' character to prevent special treatment of the '\$', '[', ']', and ',' characters.

ntrans [*inchars* [*outchars*]]

Set or unset the filename-character-translation mechanism. If you do not specify an argument, the filename-character-translation mechanism is unset. If you specify an argument, **ntrans** translates characters in remote filenames during **mput** commands and **put** commands issued without a specified remote-target filename, and translates characters in local filenames during **mget** commands and **get** commands issued without a specified local-target filename.

This command is useful when you are connecting to a non-UNIX remote computer with different file-naming conventions or practices. **ntrans** replaces characters in a filename matching a character in *inchars* with the corresponding character in *outchars*. If the character's position in *inchars* is longer than the length of *outchars*, **ntrans** deletes the character from the filename.

open *host* [*port*] Establish a connection to the specified *host* FTP server. You can specify an optional port number, in which case **ftp** attempts to contact an FTP server at that port. If the **auto-login** option is on (default), **ftp** also attempts to automatically log you in to the FTP server (see below).

prompt Toggle interactive prompting. Interactive prompting occurs during multiple file transfers to allow you to selectively retrieve or store files. If prompting is turned off (default is on), any **mget** or **mput** transfers all files, and any **mdelete** deletes all files.

proxy ftp-command

Execute an **ftp** command on a secondary control connection. This command allows you to connect simultaneously to two remote FTP servers for transferring files between them. The first **proxy** command should be an **open**, to establish the secondary control connection. Enter the command **proxy ?** to see other **ftp** commands executable on the secondary connection. The following commands behave differently when prefaced by **proxy**: **open** does not define new macros during the auto-login process, **close** does not erase existing macro definitions, **get** and **mget** transfer files from the host on the primary control connection to the host on the secondary control connection, and **put**, **mput**, and **append** transfer files from the host on the secondary control connection to the host on the primary control connection. Third-party file transfers depend upon support of the FTP protocol PASV command by the server on the secondary control connection.

put local-file [remote-file]

Store a local file on the remote machine. If you do not specify *remote-file*, **put** uses the local filename after processing according to any **ntrans** or **nmap** settings in naming the remote file. **ftp** uses the current settings for **type**, **form**, **mode**, and **structure**.

pwd

Print the name of the current working directory on the remote machine.

quit

This is a synonym for **bye**.

quote arg1 arg2 ...

This command sends the arguments you specify, verbatim, to the remote FTP server.

recv remote-file [local-file]

This is a synonym for **get**.

remotehelp [command-name]

Request help from the remote FTP server. If a *command-name* is specified it is supplied to the server as well.

rename [from] [to]

Rename the file *from* on the remote machine, to the file *to*.

reset

Clear the reply queue. This command resynchronizes command/reply sequencing with the remote FTP server. Resynchronization may be necessary following a violation of the FTP protocol by the remote server.

rmdir directory-name

Delete the specified directory on the remote machine.

runique

Toggle storing of files on the local system with unique filenames. If a file already exists with a name equal to the target local filename for a **get** or **mget** command, **runique** appends a **.1** to

the name. If the resulting name matches another existing file, **runique** appends a **.2** to the original name. If this process continues up to **.99**, **runique** prints an error message. **ftp** does not execute the transfer, and reports the generated unique filename. Note that **runique** does not affect local files generated from a shell command (see below). The default value is off.

send *local-file* [*remote-file*]

This is a synonym for **put**.

sendport

Toggle the use of PORT commands. By default, **ftp** attempts to use a PORT command when establishing a connection for each data transfer. The use of PORT commands can prevent delays when you perform multiple file transfers. If the PORT command fails, **ftp** uses the default data port. When the use of PORT commands is disabled, **ftp** does not attempt to use PORT commands for each data transfer. This is useful for certain FTP implementations that ignore PORT commands but indicate, incorrectly, that they are accepted.

status

Show the current status of **ftp**.

struct [*struct-name*]

Set the file transfer structure to *struct-name*; either **stream** or **record**. By default, **struct** uses **stream** structure.

sunique

Toggle storing of files on remote machine under unique filenames. The remote FTP server must support the FTP protocol STOU command for successful completion. The remote server reports unique names. The default value is off.

tenex

Set the file transfer type to that needed to talk to TENEX machines.

trace

Toggle packet tracing.

type [*type-name*]

Set the file transfer type to *type-name*; one of **ascii**, **binary**, **image**, or **tenex**. If you do not specify a type, **type** prints the current type. The default type is **ascii** (network ASCII).

user *user-name* [*password*] [*account*]

Identify yourself to the remote FTP server. If the password is not specified and the server requires it, **ftp** will prompt the user for it (after disabling local echo). If the FTP server requires an account field and you do not specify it, **ftp** prompts for it. If you specify an account field, **ftp** relays an account command to the remote server after the log-in sequence is completed, if the remote server did not require it for logging in. Unless you invoke **ftp** with "auto-login" disabled, **ftp** executes this process automatically, on initial connection to the FTP server.

verbose

Toggle verbose mode. In verbose mode, **ftp** displays all responses from the FTP server and also reports statistics

regarding the efficiency of a file transfer, when the transfer completes. By default, verbose is on.

? [*command*] This is a synonym for **help**.

You can enclose command arguments that have embedded spaces in quotation (") marks.

ABORTING A FILE TRANSFER

Use the terminal interrupt key (usually CTRL/C) to abort a file transfer. **ftp** immediately stops sending transfers. You can stop receiving transfers by sending a FTP protocol ABOR command to the remote server and discarding any further data received. The speed at which this is accomplished depends on the remote server's support for ABOR processing. If the remote server does not support the ABOR command, an "ftp>" prompt does not appear until the remote server has completed sending the requested file.

The terminal interrupt key sequence is ignored when **ftp** has completed any local processing and is awaiting a reply from the remote server. A long delay in this mode may result from the ABOR processing described above, or from unexpected behavior by the remote server, including violations of the FTP protocol. If the delay results from unexpected remote server behavior, you must kill the local **ftp** program by hand.

FILE-NAMING CONVENTIONS

ftp processes files that you specify as arguments according to the following rules.

- 1) If you specify the filename as a dash (-), **ftp** uses **stdin** (for reading) or **stdout** (for writing).
- 2) If the first character of the filename is "|", **ftp** interprets the remainder of the argument as a shell command, then forks a shell, using **popen(3)** with the argument you specify, and reads (writes) from **stdout** (**stdin**). If the shell command includes spaces, you must enclose the argument in quotation marks; for example, "'| ls -lt'". A particularly useful example of this mechanism is "**dir |more**".
- 3) Failing the above checks, if globbing is enabled, **ftp** expands local filenames according to the rules used in the 4=1 .BR csh (1); See the **glob** command for a comparison. If **ftp** expects a single local file (for example, **put**), it uses only the first filename generated by the "globbing" operation.
- 4) For **mget** commands and **get** commands with unspecified local filenames, the local filename is the remote filename, that a **case**, **ntrans**, or **nmap** setting can change. The remote server can then change the resulting filename, if **runique** is on.
- 5) For **mput** commands and **put** commands with unspecified remote filenames, the remote filename is the local filename, that a **ntrans** or **nmap** setting can change. The remote server can then change the resulting filename, if **sunique** is on.

FILE TRANSFER PARAMETERS

The FTP specification specifies many parameters that may affect a file transfer. The type can be one of **ascii**, **image** (binary), **ebcdic**, and **local byte size**. **ftp** supports the **ascii** and **image** types of file transfer, plus **local byte size 8** for **tenex** mode transfers.

ftp supports only the default values for the remaining file transfer parameters: **mode**, **form**, and **struct**.

OPTIONS

You can specify options on the command line, or to the command interpreter.

- v** (verbose on) Forces **ftp** to show all responses from the remote server, as well as report on data transfer statistics.
- n** Restrains **ftp** from attempting "auto-login" on initial connection. If auto-login is enabled, **ftp** checks the **.netrc** (see below) file in your home directory for an entry describing an account on the remote machine. If no entry exists, **ftp** prompts for the remote machine log-in name (the default is the user identity on the local machine), and, if necessary, prompts for a password and an account with which to log in.
- l** Turns off interactive prompting during multiple file transfers.
- d** Enables debugging.
- g** Disables filename globbing.

THE .netrc FILE

The **.netrc** file contains log-in and initialization information used by the auto-login process. It resides in your home directory. Information in the **.netrc** file consists of the following tokens, which are separated by spaces, tabs, or newlines.

Note that for this file to work, you must set up the correct ownership and permissions on the file. The file must be owned by the user of the home directory, and the file must have permissions of 600 or 660 (which gives rights only to you or members of your group). For example,

```
% ls -l .netrc
-rw-----+ 1 myself 121 Apr 5 10:46 .netrc
```

- machine name** Identify a remote machine name. The auto-login process searches the **.netrc** file for a **machine** token that matches the remote machine specified on the **ftp** command line or as an **open** command argument. Once a match is made, the subsequent **.netrc** tokens are processed, stopping when the end-of-file is reached or another **machine** token is encountered.
- login name** Identify a user on the remote machine. If this token is present, the auto-login process initiates a login using the specified **name**.
- password string** Supply a password. If this token is present, the auto-login process supplies the **string** if the remote server requires a password as part of the log-in process. Note that if this token is present in

	the .netrc file, ftp aborts the auto-login process if the .netrc is readable by anyone besides the user.
account string	Supply an additional account password. If this token is present, the auto-login process supplies the <i>string</i> if the remote server requires an additional account password, or the auto-login process initiates an ACCT command if it does not.
macrodef name	Define a macro. This token functions like the ftp macrodef command functions. A macro is defined with the specified <i>name</i> ; its contents begin with the next .netrc line and continue until a null line (consecutive new-line characters) is encountered. If a macro named init is defined, ftp automatically executes it as the last step in the auto-login process.

BUGS

Correct execution of many commands depends upon proper behavior by the remote server.

An error in the treatment of carriage returns in the 4.2BSD UNIX ASCII-mode transfer code has been corrected. This correction may result in incorrect transfers of binary files to and from 4.2BSD servers using the ASCII type. Avoid this problem by using the binary image type.

ftp does not support the options: structured page, mode block, or mode compressed.

SEE ALSO

Using TCP/IP Network Applications.

NAME

get – get a version of an SCCS file

SYNOPSIS

get [*options*] *file*

DESCRIPTION

The **get** command generates an ASCII text file from each named SCCS file according to the options that you specify in the command line. These options may appear in any order, but they apply to all named SCCS files. If you supply a directory name, **get** behaves as though each file in the directory is specified as a named file, except that it silently ignores non-SCCS and unreadable files. If you specify a dash (–) for the name of *file*, **get** reads the standard input, taking each line to be the name of an SCCS file to be processed.

The generated text is normally written into a file called the *g-file*, whose name is derived from the SCCS filename by removing the “s.” prefix.

For each file processed, **get** responds (on the standard output) with the SID being accessed and with the number of lines retrieved from the SCCS file.

OPTIONS

Each of the options is explained as though only one SCCS file is to be processed, but the effects of any option apply independently to each named file.

–rSID Specify the SCCS *ID*entification string (SID) of the version (delta) of an SCCS file to be retrieved. See Table 1 to determine what version of an SCCS file is retrieved as a function of the SID specified.

–ccutoff Indicate *cutoff* date-time in this format:

YY[MM[DD[HH[MM[SS]]]]]

Include in the ASCII text file only deltas to the SCCS that were produced up to the specified *cutoff* date-time. Units omitted from the date-time value default to their maximum possible values; that is, **–c7502** is equivalent to **–c750228235959**. Any number of non-numeric characters may separate the various two-digit pieces of the *cutoff* date-time. Thus, you can specify a *cutoff* date in the following format:

–c77/2/2 9:22:25.

You can use the **%E%** and **%U%** identification keywords for nested executions of **get** within the command line input, as in this example:

!get –c %E% %U% s.file

–e Edit or make a change (delta) to the SCCS file via a subsequent use of **delta(1)**. The SID of the delta to be made appears after the SID accessed, and before the number of lines generated. If you have more than one named file, or if you specify a directory or standard input, **get** prints each filename (preceded by a newline) before processing it.

For a particular version SID of the SCCS file, using this option prevents further executions of **get** for editing on the same SID until **delta(1)** is

executed or the **J** (joint edit) flag is set in the SCCS file. Concurrent use of this option for different SIDs is always allowed.

If the *g-file* generated by **get** with an **-e** option is accidentally ruined in the process of editing it, you can generate it by re-executing the **get** command with the **-k** option in place of the **-e** option.

SCCS file protection specified via the ceiling, floor, and authorized user list stored in the SCCS file is enforced when the **-e** option is used. See **admin(1)** for more information.

-b Provide the new delta with an SID in a new branch as shown in Table 1. This option only works when it is used with the **-e** option, the **b** flag is present in the named file, and the retrieved delta is a leaf delta (one that has no successors on the SCCS file tree). A branch delta may always be created from a non-leaf delta. Refer to **admin(1)** for further details.

-l*list* Specify a *list* of deltas to be included (forced to be applied) in the creation of the generated file. The *list* has the following syntax:

```
<list> ::= <range> | <list> , <range>
<range> ::= SID | SID - SID
```

SID, the SCCS IDentification of a delta, may be in any form shown in the "SID Specified" column of Table 1. Partial SIDs are interpreted as shown in the "SID Retrieved" column of Table 1. Included deltas are identified on the list by the notation "Included".

-x*list* Specify a *list* of deltas to be excluded (forced not to be applied) in the creation of the generated file. Excluded deltas are listed following the notation "Excluded". See the **-l** option for the *list* format.

-k Suppress replacement of identification keywords (see below) in the retrieved text by their value. The **-k** option is implied by the **-e** option.

-l[*p*] Write a delta summary into an *l-file*. If **-lp** is used, do not create an *l-file*. Instead, write the delta summary on the standard output. See the FILES section for the format of the *l-file*.

-p Write the text retrieved from the SCCS file on the standard output. Do not create a *g-file*. Send all output that normally goes to the standard output to file descriptor 2 instead, unless the **-s** option is used, in which case remove the output.

-m Precede each text line retrieved from the SCCS file by the SID of the delta that inserted the text line in the SCCS file. The format is as follows: SID, followed by a horizontal tab, followed by the text line.

-n Precede each generated text line with the **%M%** identification keyword value (see below). The format is as follows: **%M%** value, followed by a horizontal tab, followed by the text line. When both the **-m** and **-n** options are used, the format is: **%M%** value, followed by a horizontal tab, followed by the **-m** option generated format.

- s** Suppress all output normally written on the standard output. However, do not alter fatal error messages, which always go to file descriptor 2.
- g** Suppress the actual retrieval of text from the SCCS file. Used primarily to generate an *l-file* or to verify the existence of a particular SID.
- t** Access the most recently created ("top") delta in a given release (e.g., **-r1**), or release and level (e.g., **-r1.2**).
- wstring** Substitute *string* for all occurrences of "%W% of %G%" when executing a get on the named file.
- aseq-no** Specify the delta sequence number of the SCCS file delta (version) to be retrieved. See `scsfile(4)` for further details. This option is used by the `comb(1)` command; it is not a particularly useful option. If both the **-r** and **-a** options are specified, the **-a** option is used. Be careful when using the **-a** option in conjunction with the **-e** option, as the SID of the delta to be created may not be the one that you had expected. The **-r** option can be used with the **-a** and **-e** options to control the naming of the SID of the delta to be created.

Determination of SCCS Identification String

SID* Specified	-b Keyletter Used†	Other Conditions	SID Retrieved	SID of Delta to be Created
none‡	no	R defaults to mR	mR.mL	mR.(mL+1)
none‡	yes	R defaults to mR	mR.mL	mR.mL.(mB+1).1
R	no	R > mR	mR.mL	R.1***
R	no	R = mR	mR.mL	mR.(mL+1)
R	yes	R > mR	mR.mL	mR.mL.(mB+1).1
R	yes	R = mR	mR.mL	mR.mL.(mB+1).1
R	—	R < mR and R does <i>not</i> exist	hR.mL**	hR.mL.(mB+1).1
R	—	Trunk succ.# in release > R and R exists	R.mL	R.mL.(mB+1).1
R.L	no	No trunk succ.	R.L	R.(L+1)
R.L	yes	No trunk succ.	R.L	R.L.(mB+1).1
R.L	—	Trunk succ. in release ≥ R	R.L	R.L.(mB+1).1
R.L.B	no	No branch succ.	R.L.B.mS	R.L.B.(mS+1)
R.L.B	yes	No branch succ.	R.L.B.mS	R.L.(mB+1).1
R.L.B.S	no	No branch succ.	R.L.B.S	R.L.B.(S+1)
R.L.B.S	yes	No branch succ.	R.L.B.S	R.L.(mB+1).1
R.L.B.S	—	Branch succ.	R.L.B.S	R.L.(mB+1).1

* R, L, B, and S are the release, level, branch, and sequence components of the SID, respectively; m means maximum. For example, "R.mL" means the maximum level number within release R; "R.L.(mB+1).1" means the first sequence number on the new branch (i.e., maximum branch number plus one) of level L within release R. If the SID specified is of the form "R.L.", "R.L.B", or "R.L.B.S", each of the specified components must exist.

** An hR is the highest existing release that is lower than the specified, *nonexistent*, release R.

*** This is used to force creation of the first delta in a new release.

Successor.

- † The **-b** option is effective only if the **b** flag is present in the file. See **admin(1)** for further information. A dash by itself (**-**) means irrelevant.
- ‡ This case applies if the **d** (default SID) flag is not present in the file. If the **d** flag is present in the file, then the SID obtained from the **d** flag is interpreted as if it had been specified on the command line. Thus, one of the other cases in this table applies.

IDENTIFICATION KEYWORDS

Identifying information is inserted into the text retrieved from the SCCS file by replacing identification keywords with their value wherever they occur. The following keywords may be used in the text stored in an SCCS file:

Keyword	Value
%M%	Module name: either the value of the m flag in the file, or if absent, the name of the SCCS file with the leading s. removed.
%I%	SCCS Identification (SID) (%R%.%L%.%B%.%S%) of the retrieved text.
%R%	Release.
%L%	Level.
%B%	Branch.
%S%	Sequence.
%D%	Current date (YY/MM/DD).
%H%	Current date (MM/DD/YY).
%T%	Current time (HH:MM:SS).
%E%	Date newest applied delta was created (YY/MM/DD).
%G%	Date newest applied delta was created (MM/DD/YY).
%U%	Time newest applied delta was created (HH:MM:SS).
%Y%	Module type: value of the t flag in the SCCS file. See admin(1) .
%F%	SCCS filename.
%P%	Fully-qualified SCCS filename.
%Q%	The value of the q flag in the file. See admin(1) .
%C%	Current line number. This keyword is meant to identify program output messages such as "this should not have happened" when an error occurs. It should not be used on every line to provide sequence numbers.
%Z%	The four-character string @(#) recognizable by what(1) .
%W%	A shorthand notation for constructing what(1) strings for UNIX system program files. The format is: %W%~=%Z%%M%<horizontal-tab>%I%
%A%	Another shorthand notation for constructing what(1) strings for non-UNIX system program files. The format is: %A%~=%Z%%Y%~%M%~%I%%Z%

FILES

get may create several auxiliary files. These files are known generically as the *g-file*, *l-file*, *p-file*, and *z-file*. The letter before the hyphen is called the tag. An auxiliary filename is formed from the SCCS filename. The last component of all SCCS filenames must have the form *s.module-name*. The auxiliary files are named by replacing the leading **s** with the tag. The *g-file* is an exception to this scheme: the *g-*

file is named by removing the *s.* prefix. For example, the auxiliary file names of *s.xyz.c* are *xyz.c*, *l.xyz.c*, *p.xyz.c*, and *z.xyz.c*, respectively.

The *g-file*, which contains the generated text, is created in the current directory (unless the *-p* option is used). A *g-file* is created in all cases, whether or not any lines of text were generated by the *get*. It is owned by the real user. If the *-k* option is used or implied its mode is 644; otherwise, its mode is 444. Only the real user needs write permission in the current directory.

The *l-file* contains a table showing which deltas were applied in generating the retrieved text. The *l-file* is created in the current directory if the *-l* option is used; its mode is 444 and it is owned by the real user. Only the real user needs write permission in the current directory.

Lines in the *l-file* have the following format:

- a. A blank character if the delta was applied; * otherwise.
- b. A blank character if the delta was applied or was not applied and ignored; * if the delta was not applied and was not ignored.
- c. A code indicating a special reason why the delta was or was not applied. The codes are:
 - I: Included.
 - X: Excluded.
 - C: Cut off (by a *-c* option).
- d. Blank.
- e. SCCS identification (SID).
- f. Tab character.
- g. Date and time (in the form YY/MM/DD-HH:MM:SS) of creation.
- h. Blank.
- i. Log-in name of the person who created the delta.

The comments and MR data follow on subsequent lines, indented one horizontal tab character. A blank line terminates each entry.

The *p-file* passes information resulting from a *get* with an *-e* option along to *delta*. Its contents are also used to prevent a subsequent execution of *get* with an *-e* option for the same SID until *delta* is executed or the joint edit flag, *J*, is set in the SCCS file. The *p-file* is created in the directory containing the SCCS file. The effective user must have write permission in that directory. Its mode is 644 and it is owned by the effective user. The format of the *p-file* is: the gotten SID, followed by a blank, followed by the SID that the new delta will have when it is made, followed by a blank, followed by the log-in name of the real user, followed by a blank, followed by the date-time the *get* was executed, followed by a blank and the *-i* option if it was present, followed by a blank and the *-x* option if it was present, followed by a new-line. There can be an arbitrary number of lines in the *p-file* at any time; no two lines can have the same new delta SID.

The *z-file* serves as a *lock-out* mechanism against simultaneous updates. Its contents are the binary (two bytes) process ID of the command (i.e., **get**) that created it. The *z-file* is created in the directory containing the SCCS file for the duration of **get**. The same protection restrictions as those for the *p-file* apply for the *z-file*. The *z-file* is created with mode 444.

BUGS

If the effective user has write permission (either explicitly or implicitly) in the directory containing the SCCS files, but the real user does not, then only one file may be named when the **-e** option is used.

DIAGNOSTICS

Use **help(1)** for explanations.

SEE ALSO

admin(1), **delta(1)**, **help(1)**, **prs(1)**, **sccs(1)**, **what(1)**, **sccsfile(4)**

NAME

gprof – display call graph profile data

SYNOPSIS

gprof [*options*] [*objfile* [*gprofile* ...]]

DESCRIPTION

gprof produces an execution profile of C or Fortran77 programs. The effect of called routines is incorporated in the profile of each caller. The profile data is taken from the call graph profile file *gprofile* (*gmon.out* default), which is created by programs which are compiled with the **-pg** option of **cc** and **f77**. That option also links in versions of the library routines which are compiled for profiling. The symbol table in the named object file *objfile* (*a.out* default) is read and correlated with the call graph profile file. If more than one profile file is specified, the **gprof** output shows the sum of the profile information in the given profile files.

First, a flat profile is given, similar to that provided by **prof**(1). This listing gives the total execution times and call counts for each of the functions in the program, sorted by decreasing time.

Next, these times are propagated along the edges of the call graph. Cycles are discovered, and calls into a cycle are made to share the time of the cycle. A second listing shows the functions sorted according to the time they represent including the time of their call graph descendents. Below each function entry is shown its (direct) call graph children, and how their times are propagated to this function. A similar display above the function shows how this function's time and the time of its descendents is propagated to its (direct) call graph parents.

Cycles are also shown, with an entry for the cycle as a whole and a listing of the members of the cycle and their contributions to the time and call counts of the cycle.

OPTIONS

- a** Suppress the printing of statically-declared functions. If this option is given, all relevant information about the static function (e.g., time samples, calls to other functions, calls from other functions) belongs to the function loaded just before the static function in the *objfile* file.
- b** Suppress the printing of a description of each field in the profile.
- c** The static call graph of the program is discovered by a heuristic which examines the text space of the object file. Static-only parents or children are indicated with call counts of 0.
- e name** Suppress the printing of the graph profile entry for routine *name* and all its descendants (unless they have other ancestors that aren't suppressed). More than one **-e** option may be given. Only one *name* may be given with each **-e** option.
- E name** Suppress the printing of the graph profile entry for routine *name* (and its descendants) as **-e**, above, and also excludes the time spent in *name* (and its descendants) from the total and percentage time computations. (For example, **-E mcount -E mcleanup** is the default.)

- f *name*** Print the graph profile entry of only the specified routine *name* and its descendants. More than one **-f** option may be given. Only one *name* may be given with each **-f** option.
- F *name*** Print the graph profile entry of only the routine *name* and its descendants (as **-f**, above) and also uses only the times of the printed routines in total time and percentage computations. More than one **-F** option may be given. Only one *name* may be given with each **-F** option. The **-F** option overrides the **-E** option.
- s** Produce a profile file **gmon.sum** is produced which represents the sum of the profile information in all the specified profile files. This summary profile file may be given to subsequent executions of **gprof** (probably also with a **-s**) to accumulate profile data across several runs of an *objfile* file.
- z** Display routines which have zero usage (as indicated by call counts and accumulated time). This is useful in conjunction with the **-c** option for discovering which routines were never called.

FILES

a.out	the namelist and text space
gmon.out	dynamic call graph and profile
gmon.sum	summarized dynamic call graph and profile

SEE ALSO

monitor(3), profil(2), cc(1), prof(1)

"gprof: A Call Graph Execution Profiler", by Graham, S.L., Kessler, P.B., McKusick, M.K.; *Proceedings of the SIGPLAN '82 Symposium on Compiler Construction*, SIGPLAN Notices, Vol. 17, No. 6, pp. 120-126, June 1982.

BUGS

Beware of quantization errors. The granularity of the sampling is shown, but remains statistical at best. We assume that the time for each execution of a function can be expressed by the total time for the function divided by the number of times the function is called. Thus the time propagated along the call graph arcs to parents of that function is directly proportional to the number of times that arc is traversed.

Parents which are not themselves profiled will have the time of their profiled children propagated to them, but they will appear to be spontaneously invoked in the call graph listing, and will not have their time propagated further. Similarly, signal catchers, even though profiled, will appear to be spontaneous (although for more obscure reasons). Any profiled children of signal catchers should have their times propagated properly, unless the signal catcher was invoked during the execution of the profiling routine, in which case all is lost.

The profiled program must call **exit(2)** or return normally for the profiling information to be saved in the *gprofile* file.

NAME

graph – draw a graph

SYNOPSIS

graph [*option*] ...

DESCRIPTION

The **graph** command with no options takes pairs of numbers from the standard input as abscissas and ordinates of a graph. Successive points are connected by straight lines. The graph is encoded on the standard output for display by the **plot(1G)** filters.

If the coordinates of a point are followed by a nonnumeric string, that string is printed as a label beginning on the point. You can enclose labels in quotation marks "..."; if you do, the labels can be empty or contain blanks and numbers; labels never contain newlines.

OPTIONS

graph recognizes the following options, each as a separate argument.

- a** Supply abscissas automatically (they are missing from the input); spacing is given by the next argument (the default is 1). A second optional argument is the starting point for automatic abscissas (the default is 0 or a lower limit given by the **-x** option).
- b** Break (disconnect) the graph after each label in the input.
- c *string*** The character string given by *string* is the default label for each point.
- g *style*** Use the grid style, *style*: 0 for no grid, 1 for frame with ticks, 2 for full grid (2 is the default).
- l *label*** Use the string *label* as the label for the graph.
- m *mode*** Use *mode* as the mode (style) of connecting lines: 0 for disconnected, 1 for connected (1 is the default). Some devices give distinguishable line styles for other small integers.
- s** Save screen, don't erase before plotting.
- x [[[*l*] *u*] *s*]** If *l* is present, the *x* axis is logarithmic. The next one (or two) arguments are the lower (and upper) *x* limits. The third argument, if present, is the grid spacing on the *x* axis. Normally these quantities are determined automatically.
- y [[[*l*] *u*] *s*]** If *l* is present, the *y* axis is logarithmic. The next one (or two) arguments are the lower (and upper) *y* limits. The third argument, if present, is the grid spacing on the *y* axis. Normally these quantities are determined automatically.
- h *n*** Use *n* as the fraction of space for height.
- w *n*** Use *n* as the fraction of space for width.

- r *n*** Use *n* as the fraction of space to move to the right, before plotting.
- u *n*** Use *n* as the fraction of space to move up before plotting.
- t** Transpose the horizontal and vertical axes. (That is, option **-x** now applies to the vertical axis.)

graph produces a legend indicating the grid range with a grid, unless the **-s** option is present.

If a specified lower limit exceeds the upper limit, the axis is reversed.

BUGS

graph stores all points internally and drops those for which there isn't room. Segments that run out of bounds are dropped, not windowed. Logarithmic axes may not be reversed.

SEE ALSO

spline(1G), plot(1G)

NAME

grep, **egrep**, **fgrep** – search a file for a pattern

SYNOPSIS

grep [*option*] ... *expression* [*file*] ...

egrep [*option*] ... [*expression*] [*file*] ...

fgrep [*option*] ... [*strings*] [*file*] ...

DESCRIPTION

Commands of the **grep** family search the input *files* (the standard input is the default) for lines matching a pattern. Normally, each line found is copied to the standard output. Patterns used by **grep** are limited regular expressions in the style of **ex**(1); it uses a compact, non-deterministic algorithm. Patterns used by **egrep** are full regular expressions; **egrep** uses a fast, deterministic algorithm that sometimes needs exponential space. Patterns used by **fgrep** are fixed strings; it is fast and compact.

In all cases, the filename is shown if there is more than one input file. Be careful about using the **\$** * [^] () and \ characters 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 (newline-separated) *strings*.

egrep accepts extended regular expressions. In the following description *character* excludes newline:

A \ followed by a single character other than newline matches that character.

A ^ matches the beginning of a line.

A \$ matches the end of a line.

A . (period) matches any character.

A single character not otherwise endowed with special meaning matches that character.

A string enclosed in brackets [] matches any single character from the string. You can abbreviate ranges of ASCII character codes, as in a–z0–9 .

A] may occur only as the first character of the string. A literal – must be placed where it can't be mistaken for a range indicator.

A regular expression followed by an * (asterisk) matches a sequence of zero or more matches of the regular expression. A regular expression followed by a + matches a sequence of one or more matches of the regular expression. A regular expression followed by a ? matches a sequence of zero or one matches of the regular expression.

Two regular expressions concatenated match a match of the first followed by a match of the second.

Two regular expressions separated by | or newline match either a match for the first or a match for the second.

A regular expression enclosed in parentheses matches a match for the regular expression.

The order of precedence of operators at the same parenthesis level is [] then *+?, then concatenation, then | and newline.

OPTIONS

- v Print all lines but those matching.
- x Print only lines exactly matched in their entirety. (**fgrep** only).
- c Print only a count of matching lines.
- l List only the filenames with matching lines (once). Separate the names with newlines.
- n Precede each line by its relative line number in the file.
- b Precede each line by the block number on which it was found. This option helps to locate disk block numbers by context.
- i Ignore the case of letters during comparisons — that is, upper-case and lowercase are considered identical (**grep** and **fgrep** only).
- s Silent mode. Print nothing except error messages. This is useful for checking the error status.
- w Search for the expression as a word (as if surrounded by ‘\<’ and ‘\>’). See **ex(1)** for further details (**grep** only).
- e *expression* Same as a simple *expression* argument, but useful when the *expression* begins with a -.
- f *file* Take the regular expression (**egrep**) or string list (**fgrep**) from the *file*.

DIAGNOSTICS

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

BUGS

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

SEE ALSO

awk(1), ex(1), sed(1), sh(1)

NAME

groups – show group memberships

SYNOPSIS

groups [*user*]

DESCRIPTION

The **groups** command shows the groups to which you or the optionally specified *user* belong. Each user belongs to a group specified in the password file */etc/passwd* and possibly to other groups as specified in the file */etc/group*. If you don't own a file but belong to the group that owns it, you are granted group access to the file.

When a new file is created, it is given the group of the containing directory, if the directory was set up as a BSD directory (see *acl(7)*).

FILES

/etc/passwd
/etc/group

SEE ALSO

setgroups(2)

NAME

head – give the first few lines of a file

SYNOPSIS

head [*-count*] [*file ...*]

DESCRIPTION

head is a filter that gives the first *count* lines of each of the specified *files*, or of the standard input. If you omit *count*, it defaults to 10.

SEE ALSO

tail(1)

NAME

help – ask for SCCS help

SYNOPSIS

help [*arguments*]

DESCRIPTION

help finds information to explain the use of, or a particular message from, an SCCS command. You can supply several arguments, but if you supply none, **help** prompts for one.

The arguments can be message numbers (which normally appear in parentheses following messages), or command names, or one of the following types:

- Type 1 begins with non-numeric and ends in numeric. The non-numeric prefix is usually an abbreviation for the program or set of routines that produced the message. For example, **ge3** means message 3 from the **get(1)** command.
- Type 2 does not contain numerics, that is, it appears as a command name such as the name **get**.
- Type 3 is all numeric, for example, 26.

When all else fails, try **help stuck**.

FILES

/usr/lib/help	Directory containing files of message text
/usr/lib/help/helploc	File containing locations of help files not in /usr/lib/help

DIAGNOSTICS

Use **help(1)** for explanations.

SEE ALSO

admin(1), **delta(1)**, **get(1)**, **prs(1)**, **scs(1)**

NAME

hostid – set or print identifier of current host system

SYNOPSIS

hostid [*identifier*]

DESCRIPTION

The **hostid** command prints the identifier of the current host in hexadecimal. This numeric value is expected to be unique across all hosts and is commonly set to the host's Internet address. The super-user can set the host ID by giving a hexadecimal argument or the hostname; this is usually done in the start-up script **/etc/rc.local**.

SEE ALSO

gethostid(2), sethostid(2)

NAME

hostname – set or print name of current host system

SYNOPSIS

hostname [*name-of-host*]

DESCRIPTION

The **hostname** command prints the name of the current host. The super-user can set the host name by giving an argument; this is usually done in the start-up script */etc/rc.local*.

SEE ALSO

gethostname(2), *sethostname(2)*

NAME

hpc – program counter histogram

SYNOPSIS

```
hpc [-low x] [-high x]
      [-from procedure]
      [-to procedure]
      [-proc procedure]
      [-limit n] [-rate n]
      [-nhdr] [-map]
      [-brief] pathname
      [args...]
```

DESCRIPTION

hpc (**h**istogram **p**rogram **c**ounter), part of Domain/PAK (Domain Performance Analysis Kit), looks at the performance of programs at the PC level.

hpc produces a histogram of the program counter (PC) during program execution, thus helping you locate the most compute-bound portions of your program.

While your program is executing, **hpc** samples the PC at regular intervals, gathering a set of data points. Each data point records the region in which the program was executing the location of the PC when the sample was taken.

hpc divides your program into 256 equally sized regions called "buckets." The size of the region depends on the size of your program or the range you select. The smaller the region, the better the resolution of the analysis.

When execution of your program has ended, **hpc** displays statistics and a histogram (bar graph) of the PC. Each bar corresponds to an area of program memory. The length of the bar indicates how much time the program spent executing in the corresponding area. **hpc** tells you which procedures and line numbers each bar represents.

While **hpc** and your program are executing, a serial line is not available for output.

pathname (required)

Specify the name of the program to be evaluated.

args (optional)

Specify any arguments to be passed to the program *pathname*. These are not processed by **hpc**, but passed directly to your program.

Default if omitted: no arguments passed

OPTIONS

If no options are specified, a histogram is produced for the entire program, with 500 samples taken per second.

- low *x*** Specify lowest address *x* to be included in the histogram. *x* must be a hexadecimal value. If this option is omitted, the histogram starts at the beginning of the program or procedure (see **-from** below).
- high *x*** Specify highest address *x* to be included in the histogram. *x* must be a hexadecimal value. If this option is omitted, the histogram continues to the end of the program or procedure (see **-to** below).
- from *procedure*** Specify the beginning of a procedure as the lowest address to be included in the histogram. If both **-from** and **-low** are omitted, the histogram starts at the beginning of the program. Note the the procedure name is case-insensitive.
- to *procedure*** Specify the end of a procedure as the highest address to be included in the histogram. If both **-to** and **-high** are omitted, the histogram stops at the end of the program. Note the the procedure name is case-insensitive.
- proc *procedure*** Specify a single procedure to be included in the histogram. Note the the procedure name is case-insensitive.

By limiting the range of addresses in the histogram with **-low**, **-high**, **-from**, **-to**, and **-proc**, you can study a specific part of your program, such as an I/O routine.

- limit *n*** Limit the displayed histogram bars to those that represent more than *n*% of the monitored program execution. The default value for *n* is 1. Use **-limit 0** to show all histogram entries.
- rate *n*** Specify how many times *n* hpc samples the program counter per second. *n* must be a decimal number in the range 5 to 2000. The default is 500 samples per second. A higher rate results in a more accurate histogram, but tends to slow program execution.
- nhdr** Generate the histogram without the header information. Using this option makes filtering the output easier.
- map** Generate a list of the names and starting and ending locations of the procedures in the program. This list is reduced if **-from**, **-to**, **-high**, or **-low** are used to restrict monitoring to specific procedures or memory addresses. The output from this option can be quite verbose for large programs.

-brief

Produce a compact bar chart by showing only the name of the first procedure, or procedure fragment, contained in the bucket represented by each bar. By default, **dpat** shows the names of all procedures or procedure fragments contained in the bucket. This option also suppresses source-line information.

SEE ALSO

dpat(1), **dspst(1)**

NAME

indent – indent and format C program source

SYNOPSIS

indent [*input-file* [*output-file*]] [*options*]

DESCRIPTION

indent is a C program formatter. It reformats the C program in the *input-file* according to the options you specify. These are described below. They can appear before or after the filenames.

If you specify only an *input-file*, **indent** does the formatting in place, that is, it writes the formatted file back into *input-file* and writes a backup copy of *input-file* in the current directory. If *input-file* is named '/blah/blah/file', **indent** names the backup file 'file.BAK.'

If you specify *output-file*, **indent** checks to make sure it is different from *input-file*.

OPTIONS

The options listed below control the formatting style imposed by **indent**.

- bad, -nbad** If you specify **-bad**, **indent** forces a blank line after every block of declarations. The default is **-nbad**.
- bap, -nbap** If you specify **-bap**, **indent** forces a blank line after every procedure body. The default is **-nbap**.
- bbb, -nbbb** If you specify **-bbb**, **indent** forces a blank line before every block comment. The default is **-nbbb**.
- bc, -nbc** If you specify **-bc**, **indent** forces a newline after each comma in a declaration. **-nbc** turns off this option. The default is **-nbc**.
- br, -bl** If you specify **-bl**, **indent** lines up compound statements like this:

```
if (...)
{
    code
}
```

Specifying **-br** (the default) **indent** makes them look like this:

```
if (...) {
    code
}
```

- cn** The column in which comments on code start. The default is 33.
- cdn** The column in which comments on declarations start. The default is for these comments to start in the same column as those on code.
- cdb, -ncdb** Enables (**-ncdb** disables) the placement of comment delimiters on blank lines. When this option is enabled, comments look like this:

```
/*
 * this is a comment
 */
```

Rather than like this:

```
/* this is a comment */
```

This affects only block comments, not comments to the right of code. The default is **-cdb**.

- ce, -nce** Enables (**-nce** disables) forcing **else**'s to abut the immediately preceding **'}'**. The default is **-ce**.
- cin** Sets the continuation indent to be *n*. Continuation lines are indented that far from the beginning of the first line of the statement. Parenthesized expressions have extra indentation added to indicate the nesting, unless **-lp** is in effect. **-cl** defaults to the same value as **-i**.
- clin** Causes case labels to be indented *n* tab stops to the right of the containing **switch** statement. **-cli0.5** causes case labels to be indented half a tab stop. The default is **-cli0**. (This is the only option that takes a fractional argument.)
- dn** Controls the placement of comments that are not to the right of code. Specifying **-d1** means that such comments are placed one indentation level to the left of code. The default **-d0** lines up these comments with the code. See the section on comment indentation below.
- din** Specifies the indentation, in character positions, from a declaration keyword to the following identifier. The default is **-di16**.
- dj, -ndj** **-dj** left justifies declarations. **-ndj** indents declarations the same as code. The default is **-ndj**.
- ei, -nei** Enables (**-nei** disables) special **else-if** processing. If enabled, **ifs** following **elses** have the same indentation as the preceding **if** statement. The default is **-ei**.
- fc1, -nfc1** Enables (**-nfc1** disables) the formatting of comments that start in column 1. Often, comments whose leading **'/'** is in column 1 have been carefully hand formatted by the programmer. In such cases, you should use **-nfc1**. The default is **-fc1**.
- in** The number of spaces for one indentation level. The default is 8.
- lp, -nlp** Enables (**-nlp** disables) the indentation of parameter declarations from the left margin. The default is **-lp**.
- ln** Maximum length of an output line. The default is 78.

-lp, -nlp

Lines up code surrounded by parentheses in continuation lines. If a line has a left parenthesis that is not closed on that line, **indent** lines up continuation lines to start at the character position just after the left parenthesis. For example, here is how a piece of continued code looks with **-nlp** in effect:

```
p1 = first_procedure(second_procedure(p2, p3),
                    third_procedure(p4, p5));
```

With **-lp** in effect (the default) the code looks somewhat clearer:

```
p1 = first_procedure(second_procedure(p2, p3),
                    third_procedure(p4, p5));
```

Inserting two more newlines we get this:

```
p1 = first_procedure(second_procedure(p2,
                                      p3),
                    third_procedure(p4,
                                      p5));
```

-npro

Causes the profile files, **./indent.pro** and **~/indent.pro**, to be ignored.

-pcs, -npcs

If true, (**-pcs**) all procedure calls have a space inserted between the name and the **'(**. The default is **-npcs**.

-ps, -nps

If true, (**-ps**) the pointer following operator **'->'** is surrounded by spaces on either side. The default is **-nps**.

-psl, -npsl

If true (**-psl**) the names of procedures being defined are placed in column 1; their types, if any, are left on the previous lines. The default is **-psl**.

-sc, -nsc

Enables (**-nsc** disables) the placement of asterisks (*****) at the left edge of all comments. The default is **-sc**.

-sob, -nsob

If you specify **-sob**, **indent** swallows optional blank lines. You can use this to get rid of blank lines after declarations. The default is **-nsob**.

-st

Causes **indent** to take its input from **stdin**, and send its output to **stdout**.

-Ttypename

Adds *typename* to the list of type keywords. Names accumulate: you can specify **-T** more than once. You must specify all the typenames that appear in your program that are defined by **typedefs** - nothing is harmed if you miss a few, but the program won't be formatted as nicely as it should. This sounds painful, but it's really a symptom of a problem in C: **typedef** causes a syntactic change in the language and **indent** can't find all **typedefs**.

- troff** Causes **indent** to format the program for processing by **troff**. It produces a fancy listing in much the same spirit as **vgrind**. If you do not specify the output file, the default is standard output rather than formatting in place.
- v,-nv** **-v** turns on verbose mode; **-nv** turns it off. When in verbose mode, **indent** reports when it splits one line of input into two or more lines of output, and gives some size statistics at completion. The default is **-nv**.

FURTHER DESCRIPTION

You can set up your own profile of defaults to **indent** by creating a file called **.indent.pro** in either your log-in directory and/or the current directory, and including whatever options you like. Options in **.indent.pro** in the current directory override those in your log-in directory (with the exception of **-T** type definitions, which just accumulate). When you have a profile file and run **indent**, it reads the profile file to set up the program's defaults. Separate the options by spaces, tabs or newlines. However, options that you specify on the command line override options specified in the profile file.

Comments

Box comments. **indent** assumes that any comment with a dash or star immediately after the start of comment (that is, **/*-** or **/****) is a comment surrounded by a box of stars. It does not change such a comment, except that it may adjust indentation to account for the change in indentation of the first line of the comment.

Straight text. **indent** treats all other comments as straight text. **indent** fits as many words (separated by blanks, tabs, or newlines) on a line as possible. Blank lines break paragraphs.

Comment indentation

If a comment is on a line with code, it is started in the comment column, which you can set by the **-cn** command-line parameter. Otherwise, the comment is started at *n* indentation levels less than where code is currently being placed, where *n* is specified by the **-dn** command-line parameter. If the code on a line extends past the comment column, the comment starts further to the right, and the right margin may be automatically extended in extreme cases.

Preprocessor lines

In general, **indent** leaves preprocessor lines alone. The only reformatting it does is to straighten up trailing comments. It leaves embedded comments alone. **indent** recognizes conditional compilation (**#ifdef...#endif**), and attempts to compensate for the syntactic peculiarities introduced.

C syntax

indent understands a lot about C syntax, but it has a forgiving parser. It attempts to cope with the usual sorts of incomplete and misformed syntax. In particular, it handles macros such as

```
#define forever for(;;)
```

FILES

./indent.pro	Profile file
~/indent.pro	Profile file

BUGS

indent has even more options than **ls**.

A common mistake that often causes grief is typing

```
indent *.c
```

to the shell in an attempt to indent all the C programs in a directory. This is probably a bug, not a feature.

NAME

inlib – install a user-supplied library

SYNOPSIS

inlib *pathname...*

DESCRIPTION

inlib installs a library at the current shell level; it remains installed until the shell that installed it exits. See the note below for information on loading a library that is used by all processes. The newly installed library will be used to resolve external references of programs (and libraries) loaded after its installation. (Thus, previously loaded libraries and programs will not be affected.)

Note that only those global references that are marked by the binder become visible, and that the default action of the binder is to leave globals unmarked. Therefore, you should take care to mark all appropriate globals when you bind your library.

inlib is an internal shell command.

pathname (required)

Specify name of library file(s) to be installed. Multiple pathnames and wildcarding are permitted.

NAME

install – install binaries

SYNOPSIS

install [**-c**] [**-m mode**] [**-o owner**] [**-g group**] [**-s**] *binary destination*

DESCRIPTION

The **install** command moves or copies (if you specify **-c**) the *binary* to *destination*. If *destination* already exists, **install** removes it before taking any action on *binary*. If the destination is a directory, **install** moves *binary* into the destination directory with its original filename.

Unless you specify one of the appropriate options, **install** sets *destination* to 755 mode, and changes its ownership to **root** and its group identification to *staff*.

install refuses to move a file onto itself.

OPTIONS

- | | |
|-----------------|--|
| -c | Copy <i>binary</i> , instead of moving it. |
| -m mode | Specify a different <i>mode</i> . |
| -o owner | Specify a different <i>owner</i> . |
| -g group | Specify a different <i>group</i> . |
| -s | Strip <i>binary</i> after installing it. |

SEE ALSO

chgrp(1), chmod(1), cp(1), mv(1), strip(1), chown(8)

NAME

intm – install a type manager

SYNOPSIS

intm [*options*] *type_name* [*mgr_pathname*]

DESCRIPTION

intm installs a type manager for the *type_name*. The manager is copied into the type manager directory from *mgr_pathname*. If *mgr_pathname* is omitted, the file named *type_name* in the current directory is used. The **intm** command does not accept wild-cards.

type_name (required) Specify the type for which the manager is to be installed.

mgr_pathname (optional) Specify the pathname of the manager object file to install for this type.

Default if omitted: object file is named *type_name*

OPTIONS

-n *node_spec* Specify the node on which the type manager is to be installed. If this option is omitted, the type manager is installed on the current node.

-c[opy] *node-spec* Copy the type manager from the specified node.

-l List the results of the operation.

-r Replace an existing type manager if it exists.

-t[rait] *trait_name* Install the type manager for a specific trait. Without the **-trait** option you install the default type manager.

EXAMPLES

```
$ intm example_type /mydir/my_example_mgr.bin
```

```
$ intm example_type /mydir/old_example_mgr.bin -n //remote_vol -l
"/mydir/old_example_mgr.bin" installed as the manager for
type example_type on volume //remote_vol.
```

SEE ALSO

inty(1), trty(1)

NAME

inty – install a new type

SYNOPSIS

inty [*options*] *type_name* *source_volume* [*-n node_spec*]

DESCRIPTION

inty installs a type from one node to another. It installs both the type name and type manager on the target node (given by the *-n* option).

This command does not install trait-specific type managers, and does not install any type manager at all when you use the *-rait* option.

<i>type_name</i> (required)	Specify the name of the type to be installed.
<i>source_volume</i> (required)	Specify the pathname of the source volume from which to copy the type name and type manager.

OPTIONS

<i>-n node_spec</i>	Specify the node on which the type is to be installed (see <i>node_spec(5)</i>). You may also specify the entry directory of a volume mounted for software installation, as shown in the example below. If this option is omitted, the type is installed on the current node.
<i>-l</i>	List the results of the installation.
<i>-r</i>	Replace any existing type name/manager pair.
<i>-t[rait]</i>	Install a trait instead of a type.
<i>-f[ormat]</i>	Systems before SR10.1 did not support named traits. The first type you install a named trait on your system, you may see an error message showing that the types file has an older format. If you see that message, you must use the <i>-format</i> option to switch the types file to a newer version.

The new (SR10.1) format is upwards compatible with the old format, and should not interfere with older nodes. Note, however, that older nodes may mistakenly see named traits as types.

EXAMPLES

```
$ inty example_type //test_vol
Type "example_type" installed.
```

```
$ inty example_type //my_vol -n //test_vol -l
Type "example_type" installed on volume //test_vol.
```

SEE ALSO

crtty(1), *dlty(1)*, *lty(1)*, *intm(1)*, *trty(1)*, *node_spec(5)*

NAME

iso – convert files to ISO format

SYNOPSIS

```

french_to_iso   input_file output_file
german_to_iso  input_file output_file
nor.dan_to_iso input_file output_file
swedish_to_iso input_file output_file
swiss_to_iso   input_file output_file
uk_to_iso      input_file output_file

```

DESCRIPTION

These utilities convert files written with the overloaded 7-bit national fonts to the International Standards Organization (ISO) 8-bit format. The overloaded fonts include any with a specific language suffix (for example, **f7x13.french**, or **dln_f7x11.german**). If you created and/or edited a file using one of the national fonts, that file is a candidate for conversion.

You are not required to convert files, but probably will want to because

1. The overloaded fonts replace certain ASCII characters with national ones, have that subset of ASCII characters and the national characters in one file. The 8-bit fonts available as of SR10 include all the ASCII characters and the national characters.
2. The 8-bit fonts also include a wider range of national characters, so you can enter any character in any western European language. This was not always possible with the overloaded fonts. For example, there was not enough space in the overloaded font to include all the French characters, but they all exist in the 8-bit fonts.

There are two parameters to the conversion utilities. You must provide the name of the file you want to convert (*input_file*) and your *output_file*. If *output_file* already exists, the utilities abort.

The default location for the utilities is **/usr/apollo/bin**.

FILES

/usr/apollo/bin/french_to_iso	Converts overloaded French to ISO format
/usr/apollo/bin/german_to_iso	Converts overloaded German to ISO format
/usr/apollo/bin/nor.dan_to_iso	Converts overloaded Norwegian/Danish to ISO format
/usr/apollo/bin/swedish_to_iso	Converts overloaded Swedish/Finnish to ISO format
/usr/apollo/bin/swiss_to_iso	Converts overloaded Swiss to ISO format
/usr/apollo/bin/uk_to_iso	Converts overloaded U.K. English to ISO format

NAME

join – relational database operator

SYNOPSIS

join [*options*] *file1 file2*

DESCRIPTION

The **join** command forms, on the standard output, a join of the two relations specified by the lines of *file1* and *file2*. If you specify a dash ('-') for *file1*, **join** uses the standard input.

You must sort *file1* and *file2* in increasing ASCII collating sequence on the fields on which they are to be joined (normally the first in each line).

The output has one line for each pair of lines in *file1* and *file2* that have identical join fields. The output line normally consists of the common field, then the rest of the line from *file1*, and finally, the rest of the line from *file2*.

Fields are normally separated by a blank, tab or newline. **join** usually counts multiple separators as one, and discards leading separators.

OPTIONS

- an** In addition to the normal output, produce a line for each unpairable line in file *n*, where *n* is 1 or 2.
- e s** Replace empty output fields by string *s*.
- jn m** Join on the *m*th field of file *n*. If *n* is missing, use the *m*th field in each file.
- o list** Each output line comprises the fields specified in *list*, each element of which has the form *n.m*, where *n* is a file number and *m* is a field number.
- tc** Use character *c* as a separator (tab character). Every appearance of *c* in a line is significant.

EXAMPLE

To join the fourth field of each of *file1* and *file2*, type the following:

```
join -j1 4 -j2 4 file1 file2
```

BUGS

With default-field separation, the collating sequence is that of *sort -b*; with **-t**, the sequence is that of a plain sort.

The conventions of **join**, **sort**, **comm**, **uniq**, **look** and **awk** are wildly incongruous.

SEE ALSO

sort(1), **comm(1)**, **awk(1)**, **look(1)**

NAME

kbm – set/display keyboard characteristics

SYNOPSIS

```
kbm [-c args] [-d] [-l args] [-n] [-r] [-s args] [-A p_args]
      [-C p_args] [-S p_args] [-L p_args] [-R p_args] [-F p_args]
      [-T p_args]
```

DESCRIPTION

kbm allows the user to set characteristics for the keyboard. The characteristics which can be set are the shift key(s), control key(s), caps lock key(s), repeat key(s), compose key(s), and the long and short shift key(s) on the Domain multinational keyboard. In addition, autorepeat can be enable on individual keys or on sets of individual keys. Toggling (push-on, push-off) may also be set for individual keys.

The compose key is used to compose characters of the latin-1 character set which have no corresponding keys on the keyboard. Long and short shift are used to toggle the alternate key legends on the multinational keyboards.

p_args and *args* represent **key lists**. A key list is a group of function key names separated by commas. The following keys may be included in a key list:

Key Name	Positions
l1-lf	Left Function Keys
r1-r6	Right Function Keys
f0-f9	Center Function Keys
np0-np9	Numeric Pad Keys (0-9)
npa-npd	Numeric Pad Keys(/ * - +) (<i>Multinational keyboard only</i>)
npe	Numeric Pad Enter
npp	Numeric Pad Period
npf	Numeric Pad Plus (North American keyboard) or Numeric Pad Right (Multi-national keyboard)
npg	Numeric Pad Minus (North American keyboard) or Numeric Pad Left (Multi-national keyboard)
tab	TAB Key
bs	BACKSPACE Key
esc	ESCAPE Key
del	DELETE Key
ret,cr	RETURN Key
shl	Left Shift Key
shr	Right Shift Key

lock,caps	CAPS LOCK Key
ctrl	CONTROL Key (CTRL)
rept	REPEAT Key
ar,al	ALT Keys (<i>Multinational keyboard only</i>)

The following **key sets** are also defined:

Set Name Set Contents

alpha	All alphahumeric keys on the keyboard. Does not include any keys listed above.
default	Default keys relating to given option.
To disable a function, specify a key name of "none".	

In the use of compose, long shift, and short shift only, shifted keys are specified by appending the letter "s" to the key name. Control keys are appended with the letter "c" or *prefixed with a caret (^)*, while the *up* transition is marked by appending the letter "u". (For example: ar, ars, arc, ^ar, aru)

OPTIONS

If no options are specified, **kbm** displays the current keyboard type and characteristics.

-C <i>p_args</i>	Sets control keys to those specified by the list <i>p_args</i> .
-L <i>p_args</i>	Sets caps lock keys to those specified by the list <i>p_args</i> .
-M <i>p_args</i>	Sets mod1 keys to those specified by the list <i>p_args</i> . The default is Mod1: al,ar.
-S <i>p_args</i>	Sets shift keys to those specified by the list <i>p_args</i> .
-R <i>p_args</i>	Sets repeat keys to those specified by the list <i>p_args</i> .
-A <i>p_args</i>	Enables auto-repeat for those keys specified by the list <i>p_args</i> . Keys with default auto-repeat are: arrow keys (l8, la, lc, le), backspace (bs), delete (del), space(), period (.), and equal sign (=).
-F <i>p_args</i>	Enables fast delay for those keys specified by the list <i>p_args</i> . These keys must allow autorepeat. Keys with default fast delay are the arrow keys (l8, la, lc, le). The normal delay for autorepeat is approximately 0.5 seconds, while fast delay is approximately 0.25 seconds.
-T <i>p_args</i>	Enables toggle mode for those keys specified by the list <i>p_args</i> . When in toggle mode, the first press/release of the key will set the key as "down". The next press/release of the key will raise the key. Motor impaired users may find the toggle mode useful when using modifier keys such as control and shift. DO NOT set the CAPS LOCK key to toggle as the

keyboard firmware permanently implements toggle mode thereon, and enabling toggle mode will cause the key to work incorrectly. The status of the toggling CAPS LOCK key is shown by the on/off status of its LED.

- c *args*** Sets compose keys to those specified by the list *args*.
- l *args*** Sets long shift keys to those specified by the list *args*.
- s *args*** Sets short shift keys to those specified by the list *args*.
- r** Resets all keyboard settings to the appropriate default values.

compose:	none
long_alt:	als, ars
Control:	ctrl
Shift:	shl, shr
Lock:	lock
Repeat:	rept
Toggle:	none
Autorepeat:	l8, l9, lc, le, l3, bs, del, space, equal sign, period
Fast:	l9, lc, le, l3
- d** Places Model II keyboard into Model I keyboard mode. When in Model I mode, all Model II features are disabled. This option has no effect on the Model I keyboard. This mode may be useful in low-level operational situations involving the use of MD boot PROMS.
- n** Places Model II keyboard back into Model II keyboard mode.

EXAMPLES

Display current characteristics:

```
$ kbm
  keyboard:  3  North American
  compose:   none
  long_alt:  als,ars
  short_alt: al,ar
  Control:   ctrl
  Shift:     shl,shr
  Lock:      lock
  Mod1:      al,ar
  Repeat:    rept
  Toggle:    none
```

Set auto repeat on for all the alphanumeric keys and the default keys

```
$ kbm -A alpha,default
```

Set long shift keys to shifted ar and shifted al; short shift keys to al and ar.

```
$ kbm -l als,ars -s al,ar
```

Disable the compose function.

```
$ kbm -c none
```

NAME

kill – terminate or signal a process

SYNOPSIS

kill [*-sig*] *processid* ...
kill -l

DESCRIPTION

kill sends the TERM (terminate, 15) signal to the specified processes. If you specify a signal name or number preceded by '-' as the first argument, that signal is sent instead of terminate (see **sigvec**(2)). The signal names are listed by **kill -l**, and are as given in **/usr/include/signal.h**, stripped of the common SIG prefix.

The terminate signal kills processes that do not catch the signal. A **kill -9** kills any process, as the KILL signal (9) cannot be caught by conventional UNIX means. By convention, if you specify process number 0, all members in the process group (that is, processes resulting from the current login) are signaled (but beware: this works only if you use **sh**(1); not if you use **csch**(1)). Negative process numbers also have special meanings; see **kill**(2) for details.

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

The shell reports the process number of an asynchronous process started with **&**. You can also use **ps**(1) to find process numbers. **kill** is a built-in to **csch**(1); it allows job specifiers of the form "%..." as arguments, so process ID's are not as often used as **kill** arguments. See **csch**(1) for details.

NOTES

Domain/OS provides the means to catch **kill -9**.

BUGS

There is no equivalent of **kill 0** for **csch**(1) users.

There is no way to blast a process with **kill**.

SEE ALSO

csch(1), **ps**(1), **kill**(2), **sigvec**(2)

NAME

ksh – the Korn shell command programming language

SYNOPSIS

ksh [*-aefhikmnoprstuvx*] [*-o option*] . . . [*-c string*] [*-D name=val . . .*]
[*arg . . .*]

DESCRIPTION

ksh is a command programming language that executes commands read from a terminal or a file. See **Invocation** below for the meaning of arguments to the shell. **rksh** (the restricted version of this shell) is not supported by Domain/OS.

Definitions

A *metacharacter* is one of the following characters:

; & () | < > new-line space tab

A *blank* is a **tab** or a **space**. An *identifier* is a sequence of letters, digits, or underscores starting with a letter or underscore. Identifiers are used as names for *aliases*, *functions*, and *named parameters*. A *word* is a sequence of *characters* separated by one or more non-quoted *metacharacters*.

Commands

A *simple-command* is a sequence of *blank* separated words which may be preceded by a parameter assignment list. (See **Environment** below). 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 0 (see **exec(2)**). The *value* of a simple-command is its exit status if it terminates normally, or (octal) 200+*status* if it terminates abnormally (see **signal(2)** for a list of status values).

A *pipeline* is a sequence of one or more *commands* separated by **|**. The standard output of each command but the last is connected by a **pipe(2)** 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. The exit status of a pipeline is the exit status of the last command.

A *list* is a sequence of one or more pipelines separated by **;**, **&**, **&&**, or **||**, and optionally terminated by **;**, **&**, or **|&**. Of these five 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 asynchronous execution of the preceding command or pipeline with a two-way pipe established to the parent shell. The standard input and output of the spawned command can be written to and read from by the parent shell using the **-p** option of the special commands **read** and **print** described later. Only one such command can be active at any given time. The symbol **&&** (**||**) causes the *list* following it to be executed only if the preceding pipeline returns a zero (non-zero) value. An arbitrary number of new-lines may appear in a *list*, instead of semicolons, to delimit commands.

A *command* is either a simple-command or one of the following. Unless otherwise stated, the value returned by a command is that of the last simple-command executed in the command.

for *identifier* [*in word ...*] do *list* done

Each time a **for** command is executed, *identifier* 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.

select *identifier* [*in word ...*] do *list* done

A **select** command prints on standard error (file descriptor 2), the set of *words*, each preceded by a number. If *in word ...* is omitted, then the positional parameters are used instead (see **Parameter Substitution** below). The PS3 prompt is printed and a line is read from the standard input. If this line consists of the number of one of the listed *words*, then the value of the parameter *identifier* is set to the *word* corresponding to this number. If this line is empty the selection list is printed again. Otherwise the value of the parameter *identifier* is set to **null**. The contents of the line read from standard input is saved in the parameter **REPLY**. The *list* is executed for each selection until a **break** or *end-of-file* is encountered.

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 file-name generation (see **File Name 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

until *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*)

Execute *list* in a separate environment. Note, that if two adjacent open parentheses are needed for nesting, a space must be inserted to avoid arithmetic evaluation as described below. A parenthesized list used as a command argument denotes *process substitution* as described below.

{ list ;}

list is simply executed. Note that **{** is a *keyword* and requires a blank in order to be recognized.

function identifier { list ;}

identifier () { list ;}

Define a function which is referenced by *identifier*. The body of the function is the *list* of commands between **{** and **}**. (See **Functions** below).

time pipeline

The *pipeline* is executed and the elapsed time as well as the user and system time are printed on standard error.

The following keywords 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 { }
function select time

Comments

A word beginning with **#** causes that word and all the following characters up to a new-line to be ignored.

Aliasing

The first word of each command is replaced by the text of an *alias* if an *alias* for this word has been defined. The first character of an *alias* name can be any non-special printable character, but the rest of the characters must be the same as for a valid *identifier*. The replacement string can contain any valid shell script including the metacharacters listed above. The first word of each command of the replaced text will not be tested for additional aliases. If the last character of the *alias* value is a *blank* then the word following the *alias* will also be checked for *alias* substitution. Aliases can be used to redefine special builtin commands but cannot be used to redefine the keywords listed above. Aliases can be created, listed, and exported with the *alias* command and can be removed with the *unalias* command. Exported aliases remain in effect for sub-shells but must be reinitialized for separate invocations of the shell (See **Invocation** below).

Aliasing is performed when scripts are read, not while they are executed. Therefore, for an *alias* to take effect the *alias* command has to be executed before the command which references the *alias* is read.

Aliases are frequently used as a short hand for full path names. An option to the aliasing facility allows the value of the *alias* to be automatically set to the full path-name of the corresponding command. These aliases are called *tracked* aliases. The value of a *tracked* *alias* is defined the first time the corresponding command is looked up and becomes undefined each time the **PATH** variable is reset. These aliases remain *tracked* so that the next subsequent reference will redefine the value. Several *tracked* aliases are compiled into the shell. The **-h** option of the **set** command makes each command name which is a valid *alias* name into a *tracked* *alias*.

The following *exported aliases* are compiled into the shell but can be unset or redefined:

```
false='let 0'
functions='typeset -f'
history='fc -l'
integer='typeset -i'
nohup='nohup '
r='fc -e -'
true=':'
type='whence -v'
hash='alias -t'
```

Tilde Substitution

After alias substitution is performed, each word is checked to see if it begins with an unquoted `~`. If it does, then the word up to a `/` is checked to see if it matches a user name in the `/etc/passwd` file. If a match is found, the `~` and the matched login name is replaced by the login directory of the matched user. This is called a *tilde* substitution. If no match is found, the original text is left unchanged. A `~` by itself, or in front of a `/`, is replaced by the value of the `HOME` parameter. A `~` followed by a `+` or `-` is replaced by the value of the parameter `PWD` and `OLDPWD` respectively.

In addition, the value of each *keyword parameter* is checked to see if it begins with a `~` or if a `~` appears after a `:`. In either of these cases a *tilde* substitution is attempted.

Command Substitution

The standard output from a command enclosed in parenthesis preceded by a dollar sign (`$()`) or a pair of grave accents (```) may be used as part or all of a word; trailing new-lines are removed. In the second (archaic) form, the string between the quotes is processed for special quoting characters before the command is executed. (See **Quoting** below). The command substitution `$(cat file)` can be replaced by the equivalent but faster `${<file}`. Command substitution of most special commands that do not perform input/output redirection are carried out without creating a separate process.

Parameter Substitution

A *parameter* is an *identifier*, one or more digits, or any of the characters `*`, `@`, `#`, `?`, `-`, `$`, and `!`. A *named parameter* (a parameter denoted by an identifier) has a *value* and zero or more *attributes*. *Named parameters* can be assigned *values* and *attributes* by using the `typeset` special command. The attributes supported by the shell are described later with the `typeset` special command. Exported parameters pass values and attributes to sub-shells but only values to the environment.

The shell supports a limited one-dimensional array facility. An element of an array parameter is referenced by a *subscript*. A *subscript* is denoted by a `[`, followed by an *arithmetic expression* (see Arithmetic evaluation below) followed by a `]`. The value of all subscripts must be in the range of 0 through 511. Arrays need not be declared. Any reference to a named parameter with a valid subscript is legal and an array will be created if necessary. Referencing an array without a subscript is equivalent to referencing the first element.

The *value* of a *named parameter* may also be assigned by writing:

```
name=value [ name=value ] ...
```

If the integer attribute, `-l`, is set for *name* the *value* is subject to arithmetic evaluation as described below.

Positional parameters, parameters denoted by a number, may be assigned values with the `set` special command. Parameter `$0` is set from argument zero when the shell is invoked.

The character `$` is used to introduce substitutable *parameters*.

`${parameter}`

The value, if any, of the parameter is substituted. The braces are required when *parameter* is followed by a letter, digit, or underscore that is not to be interpreted as part of its name or when a named parameter is subscripted. If *parameter* is one or more digits then it is a positional parameter. A positional parameter of more than one digit must be enclosed in braces. If *parameter* is `*` or `@`, then all the positional parameters, starting with `$1`, are substituted (separated by a field separator character). If an array *identifier* with subscript `*` or `@` is used, then the value for each of the elements is substituted (separated by a field separator character).

`${#parameter}`

If *parameter* is `*` or `@`, the number of positional parameters is substituted. Otherwise, the length of the value of the *parameter* is substituted.

`${#identifier[*]}`

The number of elements in the array *identifier* is substituted.

`${parameter:-word}`

If *parameter* is set and is non-null 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 non-null then substitute its value; otherwise, print *word* and exit from the shell. If *word* is omitted then a standard message is printed.

`${parameter:+word}`

If *parameter* is set and is non-null then substitute *word*; otherwise substitute nothing.

`${parameter#pattern}`

`${parameter##pattern}`

If the shell *pattern* matches the beginning of the value of *parameter*, then the value of this substitution is the value of the *parameter* with the matched portion deleted; otherwise the value of this *parameter* is substituted. In the first form the smallest matching pattern is deleted and in the latter form the largest matching pattern is deleted.

```

${parameter%pattern}
${parameter%%pattern}

```

If the shell *pattern* matches the end of the value of *parameter*, then the value of *parameter* with the matched part deleted; otherwise substitute the value of *parameter*. In the first form the smallest matching pattern is deleted and in the latter form the largest matching pattern is deleted.

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 or not.

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 executed command.
- \$ The process number of this shell.
- The last argument of the previous command. This parameter is not set for commands which are asynchronous. This parameter is also used to hold the name of the matching **MAIL** file when checking for mail. Finally, the value of this parameter is set to the full pathname of each program the shell invokes and is passed in the *environment*.
- ! The process number of the last background command invoked.
- PPID The process number of the parent of the shell.
- PWD The present working directory set by the **cd** command.
- OLDPWD The previous working directory set by the **cd** command.
- RANDOM Each time this parameter is referenced, a random integer is generated. The sequence of random numbers can be initialized by assigning a numeric value to **RANDOM**.
- REPLY This parameter is set by the **select** statement and by the **read** special command when no arguments are supplied.
- SECONDS Each time this parameter is referenced, the number of seconds since shell invocation is returned. If this parameter is assigned a value, then the value returned upon reference will be the value that was assigned plus the number of seconds since the assignment.

The following parameters are used by the shell:

- CDPATH The search path for the **cd** command.
- COLUMNS If this variable is set, the value is used to define the width of the edit window for the shell edit modes and for printing **select** lists.

EDITOR

If the value of this variable ends in **emacs**, **gmacs**, or **vi** and the **VISUAL** variable is not set, then the corresponding option (see Special Command **set** below) will be turned on. This value should be unset for shells running in Apollo transcript pads.

ENV

If this parameter is set, then parameter substitution is performed on the value to generate the pathname of the script that will be executed when the shell is invoked. (See **Invocation** below.) This file is typically used for **alias** and **function** definitions.

FCEDIT

The default editor name for the **fc** command. In Apollo transcript pads, this variable should be set to 'pad'. On dialup lines or in VT100 windows, values like 'vi' or 'emacs' are useful.

IFS

Internal field separators, normally **space**, **tab**, and **new-line** that is used to separate command words which result from command or parameter substitution and for separating words with the special command **read**. The first character of the **IFS** parameter is used to separate arguments for the "\$*" substitution (See **Quoting** below).

HISTFILE

If this parameter is set when the shell is invoked, then the value is the pathname of the file that will be used to store the command history. (See **Command Re-entry** below.)

HISTSIZE

If this parameter is set when the shell is invoked, then the number of previously entered commands that are accessible by this shell will be greater than or equal to this number. The default is 128.

HOME The default argument (home directory) for the **cd** command.

LINES If this variable is set, the value is used to determine the column length for printing **select** lists. Select lists will print vertically until about two-thirds of **LINES** lines are filled.

MAIL If this parameter is set to the name of a mail file *and* the **MAILPATH** parameter is not set, then the shell informs the user of arrival of mail in the specified file.

MAILCHECK

This variable specifies how often (in seconds) the shell will check for changes in the modification time of any of the files specified by the **MAILPATH** or **MAIL** parameters. The default value is 600 seconds. When the time has elapsed the shell will check before issuing the next prompt.

MAILPATH

A colon (:) separated list of file names. If this parameter is set then the shell informs the user of any modifications to the specified files that have occurred within the last **MAILCHECK** seconds. Each file name can be followed by a ? and a message that will be printed. The message will undergo parameter and command substitution with the parameter, **\$_** defined as the name of the file that has changed. The default message is *you have mail in \$_*.

- PATH** The search path for commands (see **Execution** below). The user may not change **PATH** if executing under **rksh** (except in **.profile**).
- PS1** The value of this parameter is expanded for parameter substitution to define the primary prompt string which by default is "\$ ". The character ! in the primary prompt string is replaced by the *command* number (see **Command Re-entry** below).
- PS2** Secondary prompt string, by default "> ".
- PS3** Selection prompt string used in a **select** loop, by default "#? ".
- SHELL** The pathname of the *shell* is kept in the environment. This value should be unset for shells running in Apollo transcript pads.
- TMOUT** If set to a value greater than zero, the shell will terminate if a command is not entered within the prescribed number of seconds after issuing the **PS1** prompt. (Note that the shell can be compiled with a maximum bound for this value which cannot be exceeded.)
- VISUAL** If the value of this variable ends in **emacs**, **gmacs**, or **vi** then the corresponding option (see **Special Command set** below) will be turned on.

The shell gives default values to **PATH**, **PS1**, **PS2**, **MAILCHECK**, **TMOUT** and **IFS**, while **HOME**, **SHELL ENV** and **MAIL** are not set at all by the shell (although **HOME** is set by **login(1)**). On some systems **MAIL** and **SHELL** are also set by **login(1)**.

Blank Interpretation

After parameter and command substitution, the results of substitutions are scanned for the 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.

File Name Generation

Following substitution, each command *word* is scanned for the characters *, ?, and [unless the **-f** option has been **set**. If one of these characters appears then the word is regarded as a *pattern*. The word is replaced with alphabetically sorted file names that match the pattern. If no file name is found that matches the pattern, then the word is left unchanged. When a *pattern* is used for file name generation, the character . at the start of a file name or immediately following a /, as well as the character / itself, must be matched explicitly. In other instances of pattern matching the / and . are not treated specially.

- * 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. If the first character following the opening "[" is a "!" then any character not enclosed is matched. A - can be included in the character set by putting it as the first or last character.

Quoting

Each of the *metacharacters* listed above (See **Definitions** above) has a special meaning to the shell and causes termination of a word unless quoted. A character may be *quoted* (i.e., made to stand for itself) by preceding it with a backslash. The pair `\new-line` is ignored. All characters enclosed between a pair of single quote marks (''), are quoted. A single quote cannot appear within single quotes. Inside double quote marks (" "), parameter and command substitution occurs and backslash quotes the characters backslash, single quote, double quote, and dollar sign. The meaning of `$*` and `$@` is identical when not quoted or when used as a parameter assignment value or as a file name. However, when used as a command argument, `$*` is equivalent to `"$1$d$2d..."`, where *d* is the first character of the IFS parameter, whereas `$@` is equivalent to `"$1" "$2" ....` Inside grave quote marks (`) backslash quotes the characters backslash, single quote, and dollar sign. If the grave quotes occur within double quotes then backslash also quotes the character space.

The special meaning of keywords or aliases can be removed by quoting any character of the keyword. The recognition of function names or special command names listed below cannot be altered by quoting them.

Arithmetic Evaluation

An ability to perform integer arithmetic is provided with the special command `let`. Evaluations are performed using *long* arithmetic. Constants are of the form `[base#]n` where *base* is a decimal number between two and thirty-six representing the arithmetic base and *n* is a number in that base. If *base* is omitted then base 10 is used.

An internal integer representation of a *named parameter* can be specified with the `-i` option of the `typeset` special command. When this attribute is selected the first assignment to the parameter determines the arithmetic base to be used when parameter substitution occurs.

Since many of the arithmetic operators require quoting, an alternative form of the `let` command is provided. For any command which begins with a `((`, all the characters until a matching `)` are treated as a quoted expression. More precisely, `((...))` is equivalent to `let "..."`.

Prompting

When used interactively, the shell prompts with the value of `PS1` before reading a command. If at any time a new-line 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* and are *not* passed on to the invoked command. Command and parameter substitution occurs before *word* or *digit* is used except as noted below. File name generation occurs only if the pattern matches a single file and blank interpretation is not performed.

`<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. No parameter substitution, command substitution or file name generation is performed on *word*. The resulting document, called a *here-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, *\new-line* 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(2)**). Similarly for the standard output using **>&digit**.
- <&-** 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 number referred to is that specified by the digit (instead of the default 0 or 1). For example:

```
... 2>&1
```

means file descriptor 2 is to be opened for writing as a duplicate of file descriptor 1.

The order in which redirections are specified is significant. The shell evaluates each redirection in terms of the (*file descriptor*, *file*) association at the time of evaluation. For example:

```
... 1>fname 2>&1
```

first associates file descriptor 1 with file *fname*. It then associates file descriptor 2 with the file associated with file descriptor 1 (i.e. *fname*). If the order of redirections were reversed, file descriptor 2 would be associated with the terminal (assuming file descriptor 1 had been) and then file descriptor 1 would be associated with file *fname*.

If a command is followed by **&** and job control is not active, 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(7)**) is a list of name-value pairs that is passed to an executed program in the same way as a normal argument list. The names must be *identifiers* and the values are character strings. 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 and marking it *export*. Executed commands inherit the environment. If the user modifies the values of these parameters or creates new ones, using the **export** or **typeset -x** commands they become part of the environment. The environment seen by any executed

command is thus composed of any name-value pairs originally inherited by the shell, whose values may be modified by the current shell, plus any additions which must be noted in **export** or **typeset -x** commands.

The environment for any *simple-command* or function may be augmented by prefixing it with one or more parameter assignments. A parameter assignment argument is a word of the form *identifier=value*. Thus:

```
TERM=450 cmd args                and
(export TERM; TERM=450; cmd args)
```

are equivalent (as far as the above execution of *cmd* is concerned).

If the **-k** flag is set, *all* parameter assignment arguments are placed in the environment, even if they occur after the command name. The following first prints **a=b c** and then **c**:

```
echo a=b c
set -k
echo a=b c
```

Functions

The **function** keyword, described in the *Commands* section above, is used to define shell functions. Shell functions are read in and stored internally. Alias names are resolved when the function is read. Functions are executed like commands with the arguments passed as positional parameters. (See **Execution** below).

Functions execute in the same process as the caller and share all files, traps (other than **EXIT** and **ERR**) and present working directory with the caller. A trap set on **EXIT** inside a function is executed after the function completes. Ordinarily, variables are shared between the calling program and the function. However, the **typeset** special command used within a function defines local variables whose scope includes the current function and all functions it calls.

The special command **return** is used to return from function calls. Errors within functions return control to the caller.

Function identifiers can be listed with the **-f** option of the **typeset** special command. The text of functions will also be listed. Function can be undefined with the **-f** option of the **unset** special command.

Ordinarily, functions are unset when the shell executes a shell script. The **-xf** option of the **typeset** command allows a function to be exported to scripts that are executed without a separate invocation of the shell. Functions that need to be defined across separate invocations of the shell should be placed in the **ENV** file.

Jobs

If the **monitor** option of the **set** command is turned on, an interactive shell associates a *job* with each pipeline. It keeps a table of current jobs, printed by the **jobs** command, and assigns them small integer numbers. When a job is started asynchronously with **&**, the shell prints a line which looks like:

```
[1] 1234
```

indicating that the job which was started asynchronously was job number 1 and had one (top-level) process, whose process id was 1234.

This paragraph and the next require features that are not in all versions of UNIX and may not apply. If you are running a job and wish to do something else you can press **CTRL/Z** (control-Z) which sends a STOP signal to the current job. The shell will then normally indicate that the job has been 'Stopped', and print another prompt. You can then manipulate the state of this job, putting it in the background with the **bg** command, or run some other commands and then eventually bring the job back into the foreground with the foreground command **fg**. A ^Z takes effect immediately and is like an interrupt in that pending output and unread input are discarded when it is typed.

A job being run in the background will stop if it tries to read from the terminal. Background jobs are normally allowed to produce output, but this can be disabled by giving the command 'stty tostop'. If you set this tty option, then background jobs will stop when they try to produce output like they do when they try to read input.

There are several ways to refer to jobs in the shell. The character % introduces a job name. If you wish to refer to job number 1, you can name it as %1. Jobs can also be named by prefixes of the string typed in to kill or restart them. Thus, on systems that support job control, 'fg %ed' would normally restart a suspended ed(1) job, if there were a suspended job whose name began with the string 'ed'.

The shell maintains a notion of the current and previous jobs. In output pertaining to jobs, the current job is marked with a + and the previous job with a -. The abbreviation %+ refers to the current job and %- refers to the previous job. %% is also a synonym for the current job.

This shell learns immediately whenever a process changes state. It normally informs you whenever a job becomes blocked so that no further progress is possible, but only just before it prints a prompt. This is done so that it does not otherwise disturb your work.

When you try to leave the shell while jobs are running or stopped, you will be warned that 'You have stopped(running) jobs.' You may use the **Jobs** command to see what they are. If you do this or immediately try to exit again, the shell will not warn you a second time, and the stopped jobs will be terminated.

Signals

The INT and QUIT signals for an invoked command are ignored if the command is followed by & and job **monitor** option is not active. Otherwise, signals have the values inherited by the shell from its parent (but see also the **trap** command below).

Execution

Each time a command is executed, the above substitutions are carried out. If the command name matches one of the *Special Commands* listed below, it is executed within the current shell process. Next, the command name is checked to see if it matches one of the user defined functions. If it does, the positional parameters are saved and then reset to the arguments of the *function* call. When the *function* completes or issues a **return**, the positional parameter list is restored and any trap set on EXIT within the function is executed. The value of a *function* is the value of the last

command executed. A function is also executed in the current shell process. If a command name is not a *special command* or a user defined *function*, a process is created and an attempt is made to execute the command via `exec(2)`.

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 `/bin`, `/usr/bin`, and the current directory in that order). The current directory can be specified by two or more adjacent colons, or by a colon at the beginning or end of 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 a directory or an `a.out` file, it is assumed to be a file containing shell commands. A sub-shell is spawned to read it. All non-exported aliases, functions, and named parameters are removed in this case. If the shell command file doesn't have read permission, or if the `setuid` and/or `setgid` bits are set on the file, then the shell executes an agent whose job it is to set up the permissions and execute the shell with the shell command file passed down as an open file. A parenthesized command is also executed in a sub-shell without removing non-exported quantities.

Command Re-entry

The text of the last `HISTSIZE` (default 128) commands entered from a terminal device is saved in a *history* file. The file `$HOME/.sh_history` is used if the `HISTFILE` variable is not set or is not writable. A shell can access the commands of all *interactive* shells which use the same named `HISTFILE`. The special command `fc` is used to list or edit a portion of this file. The portion of the file to be edited or listed can be selected by number or by giving the first character or characters of the command. A single command or range of commands can be specified. If you do not specify an editor program as an argument to `fc` then the value of the parameter `FCEDIT` is used. If `FCEDIT` is not defined then `/bin/ed` is used. The edited command(s) is printed and re-executed upon leaving the editor. The editor name `-` is used to skip the editing phase and to re-execute the command. In this case a substitution parameter of the form `old=new` can be used to modify the command before execution. For example, if `r` is aliased to `'fc -e -'` then typing `'r bad=good c'` will re-execute the most recent command which starts with the letter `c`, replacing the first occurrence of the string `bad` with the string `good`.

In-line Editing Options

Normally, each command line entered from a terminal device is simply typed followed by a new-line ('RETURN' or 'LINE FEED'). If either the `emacs`, `gmacs`, or `vi` option is active, the user can edit the command line. To be in either of these edit modes set the corresponding option. An editing option is automatically selected each time the `VISUAL` or `EDITOR` variable is assigned a value ending in either of these option names.

The editing features require that the user's terminal accept 'RETURN' as carriage return without line feed and that a space (' ') must overwrite the current character on the screen. ADM terminal users should set the "space - advance" switch to 'space'. Hewlett-Packard series 2621 terminal users should set the straps to 'bcGHxZ etX'.

The editing modes implement a concept where the user is looking through a window at the current line. The window width is the value of **COLUMNS** if it is defined, otherwise 80. If the line is longer than the window width minus two, a mark is displayed at the end of the window to notify the user. As the cursor moves and reaches the window boundaries the window will be centered about the cursor. The mark is a > (<, *) if the line extends on the right (left, both) side(s) of the window.

The in-line editing options are not useful in Apollo transcript pads. The command input pane associated with transcript pads allows full command line editing. Setting **VISUAL** or **EDITOR** in Apollo transcript pads causes the pad to flip in and out of raw mode.

In-Line editing is very useful on dialup UP terminals or in a VT100 window where no other editing is available.

Emacs Editing Mode

This mode is entered by enabling either the **emacs** or **gmacs** option. The only difference between these two modes is the way they handle ^T. To edit, the user moves the cursor to the point needing correction and then inserts or deletes characters or words as needed. All the editing commands are control characters or escape sequences. The notation for control characters is caret (^) followed by the character. For example, ^F is the notation for control F. This is entered by pressing 'f' while holding down the 'CTRL' (control) key. The 'SHIFT' key is *not* pressed. (The notation ^? indicates the DEL (delete) key.)

The notation for escape sequences is M- followed by a character. For example, M-f (pronounced Meta f) is entered by depressing ESC (ascii 033) followed by 'f'. (M-F would be the notation for ESC followed by 'SHIFT' (capital) 'F'.)

All edit commands operate from any place on the line (not just at the beginning). Neither the "RETURN" nor the "LINE FEED" key is entered after edit commands except when noted.

^F	Move cursor forward (right) one character.
M-f	Move cursor forward one word. (The editor's idea of a word is a string of characters consisting of only letters, digits and underscores.)
^B	Move cursor backward (left) one character.
M-b	Move cursor backward one word.
^A	Move cursor to start of line.
^E	Move cursor to end of line.
^]char	Move cursor to character <i>char</i> on current line.
^X^X	Interchange the cursor and mark.
erase	(User defined erase character as defined by the stty command, usually ^H or #.) Delete previous character.
^D	Delete current character.
M-d	Delete current word.
M-^H	(Meta-backspace) Delete previous word.
M-h	Delete previous word.
M-^?	(Meta-DEL) Delete previous word (if your interrupt character is ^? (DEL, the default) then this command will not work).

~T	Transpose current character with next character in <i>emacs</i> mode. Transpose two previous characters in <i>gmacs</i> mode.
^C	Capitalize current character.
M-c	Capitalize current word.
M-l	Change the current word to lower case.
^K	Kill from the cursor to the end of the line. If given a parameter of zero then kill from the start of line to the cursor.
^W	Kill from the cursor to the mark.
M-p	Push the region from the cursor to the mark on the stack.
kill	(User defined kill character as defined by the stty command, usually ^G or @.) Kill the entire current line. If two <i>kill</i> characters are entered in succession, all kill characters from then on cause a line feed (useful when using paper terminals).
^Y	Restore last item removed from line. (Yank item back to the line.)
^L	Line feed and print current line.
^@	(Null character) Set mark.
M-	(Meta space) Set mark.
^J	(New line) Execute the current line.
^M	(Return) Execute the current line.
eof	End-of-file character, normally ^D, will terminate the shell if the current line is null.
^P	Fetch previous command. Each time ^P is entered the previous command back in time is accessed.
M-<	Fetch the least recent (oldest) history line.
M->	Fetch the most recent (youngest) history line.
^N	Fetch next command. Each time ^N is entered the next command forward in time is accessed.
^Rstring	Reverse search history for a previous command line containing <i>string</i> . If a parameter of zero is given, the search is forward. <i>String</i> is terminated by a "RETURN" or "NEW LINE". If <i>string</i> is omitted, then the next command line containing the most recent <i>string</i> is accessed. In this case a parameter of zero reverses the direction of the search.
^O	Operate - Execute the current line and fetch the next line relative to current line from the history file.
M-digits	(Escape) Define numeric parameter, the digits are taken as a parameter to the next command. The commands that accept a parameter are ^, ^F, ^B, <i>erase</i> , ^D, ^K, ^R, ^P, ^N, M-., M-_, M-b, M-c, M-d, M-f, M-h and M-H.
M-letter	Soft-key - Your alias list is searched for an alias by the name <i>letter</i> and if an alias of this name is defined, its value will be inserted on the input queue. The <i>letter</i> must not be one of the above meta-functions.
M-.	The last word of the previous command is inserted on the line. If preceded by a numeric parameter, the value of this parameter determines which word to insert rather than the last word.
M-_	Same as M-.
M-*	Attempt file name generation on the current word. An asterisk is appended if the word doesn't contain any special pattern characters.

M-ESC	Same as M-* .
M-=	List files matching current word pattern if an asterisk were appended.
^U	Multiply parameter of next command by 4.
\	Escape next character. Editing characters, the user's erase, kill and interrupt (normally ^?) characters may be entered in a command line or in a search string if preceded by a \ . The \ removes the next character's editing features (if any).
^V	Display version of the shell.

Vi Editing Mode

There are two typing modes. Initially, when you enter a command you are in the *input* mode. To edit, the user enters *control* mode by typing **ESC (033)** and moves the cursor to the point needing correction and then inserts or deletes characters or words as needed. Most control commands accept an optional repeat *count* prior to the command.

When in *vi* mode on most systems, canonical processing is initially enabled and the command will be echoed again if the speed is 1200 baud or greater and it contains any control characters or less than one second has elapsed since the prompt was printed. The **ESC** character terminates canonical processing for the remainder of the command and the user can then modify the command line. This scheme has the advantages of canonical processing with the type-ahead echoing of raw mode.

If the option **viraw** is also set, the terminal will always have canonical processing disabled. This mode is implicit for systems that do not support two alternate end of line delimiters, and may be helpful for certain terminals.

Input Edit Commands

By default the editor is in input mode.

erase	(User defined erase character as defined by the stty command, usually ^H or # .) Delete previous character.
^W	Delete the previous blank separated word.
^D	Terminate the shell.
^V	Escape next character. Editing characters, the user's erase or kill characters may be entered in a command line or in a search string if preceded by a ^V . The ^V removes the next character's editing features (if any).
\	Escape the next <i>erase</i> or <i>kill</i> character.

Motion Edit Commands

These commands will move the cursor.

[count]l	Cursor forward (right) one character.
[count]w	Cursor forward one alpha-numeric word.
[count]W	Cursor to the beginning of the next word that follows a blank.
[count]e	Cursor to end of word.
[count]E	Cursor to end of the current blank delimited word.
[count]h	Cursor backward (left) one character.
[count]b	Cursor backward one word.
[count]B	Cursor to preceding blank separated word.

[count]fc	Find the next character <i>c</i> in the current line.
[count]Fc	Find the previous character <i>c</i> in the current line.
[count]tc	Equivalent to f followed by h .
[count]Tc	Equivalent to F followed by l .
;	Repeats the last single character find command, f , F , t , or T .
,	Reverses the last single character find command.
0	Cursor to start of line.
^	Cursor to first non-blank character in line.
\$	Cursor to end of line.

Search Edit Commands

These commands access your command history.

[count]k	Fetch previous command. Each time k is entered the previous command back in time is accessed.
[count]-	Equivalent to k .
[count]j	Fetch next command. Each time j is entered the next command forward in time is accessed.
[count]+	Equivalent to j .
[count]G	The command number <i>count</i> is fetched. The default is the least recent history command.
/string	Search backward through history for a previous command containing <i>string</i> . <i>String</i> is terminated by a "RETURN" or "NEW LINE". If <i>string</i> is null the previous string will be used:
?string	Same as / except that search will be in the forward direction.
n	Search for next match of the last pattern to / or ? commands.
N	Search for next match of the last pattern to / or ? , but in reverse direction. Search history for the <i>string</i> entered by the previous / command.

Text Modification Edit Commands

These commands will modify the line.

a	Enter input mode and enter text after the current character.
A	Append text to the end of the line. Equivalent to \$a .
[count]cmotion	
c[count]motion	Delete current character through the character that <i>motion</i> would move the cursor to and enter input mode. If <i>motion</i> is c , the entire line will be deleted and input mode entered.
C	Delete the current character through the end of line and enter input mode. Equivalent to c\$.
S	Equivalent to cc .
D	Delete the current character through the end of line. Equivalent to d\$.
[count]dmotion	
d[count]motion	Delete current character through the character that <i>motion</i> would move to. If <i>motion</i> is d , the entire line will be deleted.
i	Enter input mode and insert text before the current character.

I	Insert text before the beginning of the line. Equivalent to the two character sequence <code>^I</code> .
<code>[count]P</code>	Place the previous text modification before the cursor.
<code>[count]p</code>	Place the previous text modification after the cursor.
R	Enter input mode and replace characters on the screen with characters you type overlay fashion.
<code>rc</code>	Replace the current character with <i>c</i> .
<code>[count]x</code>	Delete current character.
<code>[count]X</code>	Delete preceding character.
<code>[count].</code>	Repeat the previous text modification command.
<code>~</code>	Invert the case of the current character and advance the cursor.
<code>[count]_</code>	Causes the <i>count</i> word of the previous command to be appended and input mode entered. The last word is used if <i>count</i> is omitted.
*	Causes an <i>*</i> to be appended to the current word and file name generation attempted. If no match is found, it rings the bell. Otherwise, the word is replaced by the matching pattern and input mode is entered.

Other Edit Commands

Miscellaneous commands.

`[count]ymotion`

`y[count]motion`

Yank current character through character that *motion* would move the cursor to and puts them into the delete buffer. The text and cursor are unchanged.

Y Yanks from current position to end of line. Equivalent to `y$`.

u Undo the last text modifying command.

U Undo all the text modifying commands performed on the line.

`[count]v` Returns the command `fc -e ${VISUAL:-${EDITOR:-vi}}` *count* in the input buffer. If *count* is omitted, then the current line is used.

`^L` Line feed and print current line. Has effect only in control mode.

`^J` (New line) Execute the current line, regardless of mode.

`^M` (Return) Execute the current line, regardless of mode.

Sends the line after inserting a *#* in front of the line and after each new-line. Useful for causing the current line to be inserted in the history without being executed.

= List the file names that match the current word if an asterisk were appended it.

`@letter` Your alias list is searched for an alias by the name *_letter* and if an alias of this name is defined, its value will be inserted on the input queue for processing.

Special Commands

The following simple-commands are executed in the shell process. Input/Output redirection is permitted. Unless otherwise indicated, the output is written on file

descriptor 1. Commands that are preceded by one or two `†` are treated specially in the following ways:

1. Parameter assignment lists preceding the command remain in effect when the command completes.
2. They are executed in a separate process when used within command substitution.
3. Errors in commands preceded by `††` cause the script that contains them to abort.

`† : [ arg ... ]`

The command only expands parameters. A zero exit code is returned.

`†† .file [ arg ... ]`

Read and execute commands from *file* and return. The commands are executed in the current shell environment. The search path specified by `PATH` is used to find the directory containing *file*. If any arguments *arg* are given, they become the positional parameters. Otherwise the positional parameters are unchanged.

`alias [ -tx ] [ name[ =value ] ... ]`

Alias with no arguments prints the list of aliases in the form *name=value* on standard output. An *alias* is defined for each name whose *value* is given. A trailing space in *value* causes the next word to be checked for alias substitution. The `-t` flag is used to set and list tracked aliases. The value of a tracked alias is the full pathname corresponding to the given *name*. The value becomes undefined when the value of `PATH` is reset but the aliases remained tracked. Without the `-t` flag, for each *name* in the argument list for which no *value* is given, the name and value of the alias is printed. The `-x` flag is used to set or print exported aliases. An exported alias is defined across sub-shell environments. *Alias* returns true unless a *name* is given for which no alias has been defined.

`bg [ %job ]`

This command is only built-in on systems that support job control. Puts the specified *job* into the background. The current job is put in the background if *job* is not specified.

`break [ n ]`

Exit from the enclosing **for** **while** **until** or **select** loop, if any. If *n* is specified then break *n* levels.

`continue [ n ]`

Resume the next iteration of the enclosing **for** **while** **until** or **select** loop. If *n* is specified then resume at the *n*-th enclosing loop.

`† cd [ arg ]`

`† cd old new`

This command can be in either of two forms. In the first form it changes the current directory to *arg*. If *arg* is `-` the directory is changed to the previous directory. The shell parameter `HOME` is the default *arg*. The parameter `PWD` is set to the current directory. The shell parameter `CDPATH` defines

the search path for the directory containing *arg*. Alternative directory names are separated by a colon (:). The default path is *<null>* (specifying the current directory). 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 *arg* begins with a / then the search path is not used. Otherwise, each directory in the path is searched for *arg*.

The second form of **cd** substitutes the string *new* for the string *old* in the current directory name, **PWD** and tries to change to this new directory.

The **cd** command may not be executed by **rksh**.

echo [*arg* ...]

See **echo**(1) for usage and description.

†† **eval** [*arg* ...]

The arguments are read as input to the shell and the resulting command(s) executed.

†† **exec** [*arg* ...]

If *arg* is given, the command specified by the arguments is executed in place of this shell without creating a new process. Input/output arguments may appear and affect the current process. If no arguments are given the effect of this command is to modify file descriptors as prescribed by the input/output redirection list. In this case, any file descriptor numbers greater than 2 that are opened with this mechanism are closed when invoking another program.

exit [*n*]

Causes the 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 except for a shell which has the **ignoreeof** option (See **set** below) turned on.

†† **export** [*name* ...]

The given *names* are marked for automatic export to the *environment* of subsequently-executed commands.

†† **fc** [**-e** *ename*] [**-nlr**] [*first*] [*last*]

†† **fc** **-e** - [*old=new*] [*command*]

In the first form, a range of commands from *first* to *last* is selected from the last **HISTSIZE** commands that were typed at the terminal. The arguments *first* and *last* may be specified as a number or as a string. A string is used to locate the most recent command starting with the given string. A negative number is used as an offset to the current command number. If the flag **-l** is selected, the commands are listed on standard output. Otherwise, the editor program *ename* is invoked on a file containing these keyboard commands. If *ename* is not supplied, then the value of the parameter **FCEDIT** (default **/bin/ed**) is used as the editor. When editing is complete, the edited command(s) is executed. If *last* is not specified then it will be set to *first*. If *first* is not specified the default is the previous command for editing and **-16**

for listing. The flag `-r` reverses the order of the commands and the flag `-n` suppresses command numbers when listing. In the second form the *command* is re-executed after the substitution *old=new* is performed.

fg [*%job*]

This command is only built-in on systems that support job control. If *job* is specified it brings it to the foreground. Otherwise, the current job is brought into the foreground.

inlib *pathname*

Install a user-supplied library specified by *pathname* in the current (shell) process. The library is used to resolve external references of programs (and libraries) loaded after its installation. Note that the library is not loaded into the address space unless it is needed to resolve an external reference. The list of inlibed libraries is passed to all children of the current shell. Use `llib(1)` to examine this list.

jobs [`-l`]

Lists the active jobs; given the `-l` options lists process id's in addition to the normal information.

kill [`-sig`] *process* ...

Sends either the TERM (terminate) signal or the specified signal to the specified jobs or processes. Signals are either given by number or by names (as given in `/usr/include/signal.h`, stripped of the prefix "SIG"). The signal numbers and names are listed by 'kill -l'. If the signal being sent is TERM (terminate) or HUP (hangup), then the job or process will be sent a CONT (continue) signal if it is stopped. The argument *process* can be either a process id or a job.

let *arg* ...

Each *arg* is an *arithmetic expression* to be evaluated. All calculations are done as long integers and no check for overflow is performed. Expressions consist of constants, named parameters, and operators.

The following set of operators, listed in order of decreasing precedence, have been implemented:

-	unary minus
!	logical negation
* / %	multiplication, division, remainder
+ -	addition, subtraction
<= >= < >	comparison
== !=	equality inequality
=	arithmetic replacement

Sub-expressions in parentheses () are evaluated first and can be used to override the above precedence rules. The evaluation within a precedence group is from right to left for the = operator and from left to right for the others.

A parameter name must be a valid *identifier*. When a parameter is encountered, the value associated with the parameter name is substituted and

expression evaluation resumes. Up to nine levels of recursion are permitted. The return code is 0 if the value of the last expression is non-zero, and 1 otherwise.

†† **newgrp** [*arg* ...]

Equivalent to **exec newgrp arg**

print [**-Rnprsu**[*n*]] [*arg* ...]

The shell output mechanism. With no flags or with flag **-**, the arguments are printed on standard output as described by **echo**(1). In raw mode, **-R** or **-r**, the escape conventions of **echo** are ignored. The **-R** option will print all subsequent arguments and options other than **-n**. The **-p** option causes the arguments to be written onto the pipe of the process spawned with **|&** instead of standard output. The **-s** option causes the arguments to be written onto the history file instead of standard output. The **-u** flag can be used to specify a one digit file descriptor unit number *n* on which the output will be placed. The default is 1. If the flag **-n** is used, no **new-line** is added to the output.

pwd Equivalent to **print -r - \$PWD**

read [**-prsu**[*n*]] [*name?prompt*] [*name* ...]

The shell input mechanism. One line is read and is broken up into words using the characters in IFS as separators. In raw mode, **-r**, a **** at the end of a line does not signify line continuation. 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 **-p** option causes the input line to be taken from the input pipe of a process spawned by the shell using **|&**. If the **-s** flag is present, the input will be saved as a command in the history file. The flag **-u** can be used to specify a one digit file descriptor unit to read from. The file descriptor can be opened with the **exec** special command. The default value of *n* is 0. If *name* is omitted then **REPLY** is used as the default *name*. The return code is 0 unless an end-of-file is encountered. An end-of-file with the **-p** option causes cleanup for this process so that another can be spawned. If the first argument contains a **?**, the remainder of this word is used as a *prompt* when the shell is interactive. If the given file descriptor is open for writing and is a terminal device then the prompt is placed on this unit. Otherwise the prompt is issued on file descriptor 2. The return code is 0 unless an end-of-file is encountered.

†† **readonly** [*name* ...]

The given *names* are marked readonly and these names cannot be changed by subsequent assignment.

†† **return** [*n*]

Causes a shell *function* to return to the invoking script with the return status specified by *n*. If *n* is omitted then the return status is that of the last command executed. If **return** is invoked while not in a *function* or a **.** script, then it is the same as an **exit**.

rootnode [*arg*]

Change the current node entry directory to *arg*.

set [**-aefhkmnostuvx**] [**-o option ...**] [*arg ...*]

The flags for this command have meaning as follows:

- a** All subsequent parameters that are defined are automatically exported.
- e** If the shell is non-interactive and if a command fails, execute the **ERR** trap, if set, and exit immediately. This mode is disabled while reading profiles.
- f** Disables file name generation.
- h** Each command whose name is an *identifier* becomes a tracked alias when first encountered.
- k** All parameter assignment arguments are placed in the environment for a command, not just those that precede the command name.
- m** Background jobs will run in a separate process group and a line will print upon completion. The exit status of background jobs is reported in a completion message. On systems with job control, this flag is turned on automatically for interactive shells.
- n** Read commands but do not execute them. Ignored for interactive shells.
- o** The following argument can be one of the following option names:
 - allexport** Same as **-a**.
 - errexit** Same as **-e**.
 - bgnice** All background jobs are run at a lower priority.
 - emacs** Puts you in an **emacs** style in-line editor for command entry.
 - gmacs** Puts you in a **gmacs** style in-line editor for command entry.
 - ignoreeof** The shell will not exit on end-of-file. The command **exit** must be used.
 - keyword** Same as **-k**.
 - markdirs** All directory names resulting from file name generation have a trailing **/** appended.
 - monitor** Same as **-m**.
 - noexec** Same as **-n**.
 - noglob** Same as **-f**.
 - nounset** Same as **-u**.
 - protected** Same as **-p**.

- verbose** Same as **-v**.
- trackall** Same as **-h**.
- vi** Puts you in insert mode of a **vi** style in-line editor until you hit escape character 033. This puts you in move mode. A return sends the line.
- viraw** Each character is processed as it is typed in **vi** mode.
- xtrace** Same as **-x**.
- If no option name is supplied then the current option settings are printed.
- p** Resets the **PATH** variable to the default value, disables processing of the **\$HOME/.profile** file and uses the file **/etc/suid_profile** instead of the **ENV** file. This mode is automatically enabled whenever the effective uid (gid) is not equal to the real uid (gid).
- s** Sort the positional parameters.
- t** Exit after reading and executing one command.
- u** Treat unset parameters as an error when substituting.
- v** Print shell input lines as they are read.
- x** Print commands and their arguments as they are executed.
- Turns off **-x** and **-v** flags and stops examining arguments for flags.
- Do not change any of the flags; useful in setting **\$1** to a value beginning with **-**. If no arguments follow this flag then the positional parameters are unset.

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 on the standard output.

† shift [*n*]

The positional parameters from **\$n+1 ...** are renamed **\$1 ...**, default **n** is 1. The parameter **n** can be any arithmetic expression that evaluates to a non-negative number less than or equal to **\$#**.

test [*expr*]

Evaluate conditional expression *expr*. See **test(1)** for usage and description. The arithmetic comparison operators are not restricted to integers. They allow any arithmetic expression. Four additional primitive expressions are allowed:

- L file** True if *file* is a symbolic link.
- file1 -nt file2** True if *file1* is newer than *file2*.
- file1 -ot file2** True if *file1* is older than *file2*.
- file1 -ef file2** True if *file1* has the same device and i-node number as *file2*.

times Print the accumulated user and system times for the shell and for processes run from the shell.

trap [*arg*] [*sig*] ...

arg is a command to be read and executed when the shell receives signal(s)

sig. (Note that *arg* is scanned once when the trap is set and once when the trap is taken.) Each *sig* can be given as a number or as the name of the signal. Trap commands are executed in order of signal number. Any attempt to set a trap on a signal that was ignored on entry to the current shell is ineffective. If *arg* is omitted or is `-`, then all trap(s) *sig* 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 *sig* is `ERR` then *arg* will be executed whenever a command has a non-zero exit code. This trap is not inherited by functions. If *sig* is `0` or `EXIT` and the `trap` statement is executed inside the body of a function, then the command *arg* is executed after the function completes. If *sig* is `0` or `EXIT` for a `trap` set outside any function 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.

†† `typeset [ -HLRZfilprtux[ n ][ name[ =value ] ] ... ]`

When invoked inside a function, a new instance of the parameter *name* is created. The parameter value and type are restored when the function completes. The following list of attributes may be specified:

- H This flag provides UNIX to host-name file mapping on non-UNIX machines.
- L Left justify and remove leading blanks from *value*. If *n* is non-zero it defines the width of the field, otherwise it is determined by the width of the value of first assignment. When the parameter is assigned to, it is filled on the right with blanks or truncated, if necessary, to fit into the field. Leading zeros are removed if the `-Z` flag is also set. The `-R` flag is turned off.
- R Right justify and fill with leading blanks. If *n* is non-zero it defines the width of the field, otherwise it is determined by the width of the value of first assignment. The field is left filled with blanks or truncated from the end if the parameter is reassigned. The `L` flag is turned off.
- Z Right justify and fill with leading zeros if the first non-blank character is a digit and the `-L` flag has not been set. If *n* is non-zero it defines the width of the field, otherwise it is determined by the width of the value of first assignment.
- f The names refer to function names rather than parameter names. No assignments can be made and the only other valid flags are `-t`, which turns on execution tracing for this function and `-x`, to allow the function to remain in effect across shell procedures executed in the same process environment.
- i Parameter is an integer. This makes arithmetic faster. If *n* is non-zero it defines the output arithmetic base, otherwise the first assignment determines the output base.
- l All upper-case characters converted to lower-case. The upper-case flag, `-u` is turned off.
- p The output of this command, if any, is written onto the two-way pipe.

- r The given *names* are marked readonly and these names cannot be changed by subsequent assignment.
- t Tags the named parameters. Tags are user definable and have no special meaning to the shell.
- u All lower-case characters are converted to upper-case characters. The lower-case flag, -l is turned off.
- x The given *names* are marked for automatic export to the *environment* of subsequently-executed commands.

Using + rather than - causes these flags to be turned off. If no *name* arguments are given but flags are specified, a list of *names* (and optionally the *values*) of the *parameters* which have these flags set is printed. (Using + rather than - keeps the values to be printed.) If no *names* and flags are given, the *names* and *attributes* of all *parameters* are printed.

ulimit [-acdfmpst] [*n*]

- a Lists all of the current resource limits.
- d imposes a size limit of *n* kbytes on the size of the data area.
- f imposes a size limit of *n* 512 byte blocks on files written by child processes (files of any size may be read).
- m imposes a soft limit of *n* kbytes on the size of physical memory.
- p changes the pipe size to *n* (UNIX/RT only).
- s imposes a size limit of *n* kbytes on the size of the stack area.
- t imposes a time limit of *n* seconds to be used by each process.

If no option is given, -f is assumed. If *n* is not given the current limit is printed.

umask [*nnn*]

The user file-creation mask is set to *nnn* (see **umask(2)**). If *nnn* is omitted, the current value of the mask is printed.

unalias *name* ...

The parameters given by the list of *names* are removed from the *alias* list.

unset [-f] *name* ...

The parameters given by the list of *names* are unassigned, i. e., their values and attributes are erased. Readonly variables cannot be unset. If the flag, -f, is set, then the names refer to *function* names.

ver [*systype*[*command*]]

With no arguments, return the current value of the SYSTYPE environment variable that specifies the version of UNIX commands being executed by the shell. With a *systype* argument, change the SYSTYPE environment variable to either **bsd4.3** or **sys5.3**, depending on which is specified.

wait [*n*]

Wait for the specified child process and report its termination status. If *n* is not given then all currently active child processes are waited for. The return code from this command is that of the process waited for.

whence [*-v*] *name* ...

For each *name*, indicate how it would be interpreted if used as a command name.

The flag *-v* produces a more verbose report.

Invocation.

If the shell is invoked by `exec(2)`, and the first character of argument zero (`$0`) is `-`, then the shell is assumed to be a *login* shell and commands are read from `/etc/profile` and then from either `.profile` in the current directory or `$HOME/.profile`, if either file exists. Next, commands are read from the file named by performing parameter substitution on the value of the environment parameter `ENV` if the file exists. If the `-s` flag is not present and *arg* is, then a path search is performed on the first *arg* to determine the name of the script to execute. The script *arg* must have read permission and any `setuid` and `getgid` settings will be ignored. Commands are then read as described below; the following flags are interpreted by the shell when it is invoked:

- `-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. Shell output, except for the output of the *Special commands* listed above, 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 (as told by `ioctl(2)`) then this shell is *interactive*. In this case `TERM` is ignored (so that `kill 0` does not kill an interactive shell) and `INTR` 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.
- `-Dname=value` You can use the `-D` option to specify a parameter *name* that will be set to *value* and then passed into the shell's environment. This Domain/OS option is useful for tailoring the environment of a shell invoked from a program that is not another shell (such as the Display Manager). If the `ENV` parameter is given in this way, the startup script it specifies will be run. Note that any number of `-D` options can be specified.

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 non-zero exit status. Otherwise, the shell returns the exit status of the last command executed (see also the `exit` command above). If the shell is being used non-interactively then execution of the shell file is abandoned. Runtime errors detected by the shell are reported by printing the command or function name and the error condition. If the line number that the error occurred on is greater than one, then the line number is also printed in square brackets (`[]`) after the command or function name.

CAVEATS

If a command which is a *tracked alias* is executed, and then a command with the same name is installed in a directory in the search path before the directory where the

original command was found, the shell will continue to **exec** the original command. Use the **-t** option of the **alias** command to correct this situation.

Some very old shell scripts contain a **^** as a synonym for the pipe character **|**.

If a command is piped into a shell command, then all variables set in the shell command are lost when the command completes.

Using the **fc** built-in command within a compound command will cause the whole command to disappear from the history file.

The built-in command **.file** reads the whole file before any commands are executed. Therefore, **alias** and **unalias** commands in the file will not apply to any functions defined in the file.

FILES

/etc/passwd
/etc/profile
/etc/suid_profile
\$HOME/.profile
/tmp/sh*
/dev/null

SEE ALSO

cat(1), cd(1), echo(1), test(1), vi(1), dup(2), exec(2), fork(2), ioctl(2), lseek(2), pipe(2), umask(2), wait(2), rand(3), signal(3), a.out(5), environ(7).

NAME

las – list objects mapped into the address space

SYNOPSIS

las [*options*]

DESCRIPTION

las produces a list of objects mapped into the address space. Information printed includes the virtual address range, the starting address within the object, and its pathname if available (in that order).

This command is most useful for system-level debugging.

OPTIONS

If no options are specified, **las** lists the address space of the current process.

- all** List all address space, including that occupied by Aegis.
- f[rom] address** Begin listing at the hexadecimal *address* specified.
- t[o] address** End listing at the hexadecimal *address* specified.

EXAMPLES

1.

```
$ las
```

VA Range	Obj Start	Pathname
8000 - 17FFF	0	/sys/node_data/global_data
18000 - 2FFFF	0	/lib/pmlib
30000 - 37FFF	0	/lib/syslib.peb
38000 - 4FFFF	0	/lib/kslib
50000 - 57FFF	0	/lib/trait_type_lib
58000 - 67FFF	10000	/sys/node_data/global_data
68000 - 9FFFF	0	/lib/streams
A0000 - A7FFF	0	/lib/vfmt_streams
A8000 - BFFFF	0	/lib/error
C0000 - E7FFF	0	/lib/swtlib
E8000 - F7FFF	0	/lib/ftnlib
F8000 - FFFFF	0	/lib/pbulib
100000 - 127FFF	0	/lib/gprlib
128000 - 14FFFF	0	/lib/clib
150000 - 157FFF	0	/lib/lisp_initlib
158000 - 15FFFF	0	/sys/node_data/global_rws
160000 - 16FFFF	20000	/sys/node_data/global_data
170000 - 187FFF	0	/lib/shlib
188000 - 19FFFF	0	/lib/tfp
1A0000 - 1BFFFF	0	/lib/dialoglib
1C0000 - 1C7FFF	0	/sys/node_data/ipc_data

1D0000 -	1D7FFF	30000	/sys/node_data/global_data
200000 -	2AFFFF	0	-- temporary file --
2B0000 -	2B7FFF	0	/sys/node_data/dm_mbx
2B8000 -	2BFFFF	0	/com/sh
2C0000 -	2C7FFF	0	-- temporary file --
2C8000 -	2CFFFF	0	/com/las
2D0000 -	2F7FFF	B0000	-- temporary file --
BC0000 -	BCFFFF	0	/help_area/worksite
BD0000 -	BDFFFF	0	/jtj

2944 KB mapped.

2.

\$ las -from 188000

VA Range	Obj Start	Pathname
188000 -	19FFFF	0 /lib/tfp
1A0000 -	1BFFFF	0 /lib/dialoglib
1C0000 -	1C7FFF	0 /sys/node_data/ipc_data
1D0000 -	1D7FFF	30000 /sys/node_data/global_data
200000 -	2AFFFF	0 -- temporary file --
2B0000 -	2B7FFF	0 /sys/node_data/dm_mbx
2B8000 -	2BFFFF	0 /com/sh
2C0000 -	2C7FFF	0 -- temporary file --
2C8000 -	2CFFFF	0 /com/las
2D0000 -	2F7FFF	B0000 -- temporary file --
BC0000 -	BCFFFF	0 /help_area/worksite
BD0000 -	BDFFFF	0 /jtj

1408 KB mapped.

3.

\$ las -f 188000 -t 200000

VA Range	Obj Start	Pathname
188000 -	19FFFF	0 /lib/tfp
1A0000 -	1BFFFF	0 /lib/dialoglib
1C0000 -	1C7FFF	0 /sys/node_data/ipc_data
1D0000 -	1D7FFF	30000 /sys/node_data/global_data

288 KB mapped.

NAME

last – indicate last logins of users and teletypes

SYNOPSIS

last [*-N*] [*name* ...] [*tty* ...]

DESCRIPTION

last looks in the **wtmp** file (which records all logins and logouts) for information about a user, a teletype, or any group of users and teletypes. Arguments specify users' names or teletypes of interest. You can specify teletype names fully, or abbreviate them. For example, "last 0" is the same as "last tty0". If you specify multiple arguments, **last** prints the information that applies to any of the arguments.

last prints the sessions of the specified users and teletypes, most recent first, indicating the time the session began, the duration of the session, and the teletype the session took place on. If the session is still continuing or was cut short by a reboot, **last** so indicates.

The pseudo-user **reboot** logs in at reboots of the system, thus

last reboot

gives an indication of mean time between reboots.

If you don't specify an argument, **last** prints a record of all logins and logouts, in reverse order. Specify *-N* to limit the report to *N* lines.

If **last** is interrupted, it indicates how far the search has progressed in **wtmp**. If interrupted with a quit signal (usually generated by a CTRL-\) **last** indicates how far the search has progressed so far, and continues the search.

EXAMPLES

The following command lists all of **root**'s sessions as well as all sessions on the console terminal.

last root console

FILES

/usr/adm/wtmp	Log-in database
/usr/adm/shutdownlog	Records shutdowns and the reasons for them

SEE ALSO

lastcomm(1), wtmp(5), ac(8)

NAME

lastcomm – show last commands executed in reverse order

SYNOPSIS

lastcomm [*command*] ... [*username*] ... [*terminal*] ...

DESCRIPTION

lastcomm gives information on previously executed commands. With no arguments, **lastcomm** prints information about all the commands recorded during the current accounting file's lifetime. If called with arguments, **lastcomm** prints only accounting entries with a matching command name, username, or terminal name. For example,

lastcomm a.out root ttyd0

produces a listing of all the executions of commands named **a.out** by user **root** on the terminal **ttyd0**.

For each process entry, **lastcomm** prints the following:

- The name of the user who ran the process
- Flags, as accumulated by the accounting facilities in the system
- The command name under which the process was called
- The amount of CPU time (in seconds) the process used
- The time the process exited

The flags are encoded as follows: S indicates the command was executed by the super-user, F indicates command ran after a fork, but without a following **exec**, D indicates the command terminated with a stack dump, and X indicates the command was terminated with a signal.

FILES

/usr/adm/acct

SEE ALSO

last(1), **tb(1)**, **sigvec(2)**, **acct(8)**

NAME

lbr2ar – convert lbr libraries to SR10 archive libraries

SYNOPSIS

lbr2ar [-y *dirname*] *lbrfile arfile*

DESCRIPTION

The **lbr2ar** command converts pre-SR10 lbr library files containing object modules in OBJ format to SR10 ar library archive files containing object modules in COFF format. The **lbr2ar** command extracts each object module from the *lbrfile*, executes the **obj2coff** converter to convert them to COFF, and creates a library archive (*arfile*) containing the converted object modules. Note that both the library format and the format of the individual object modules are changed.

OPTIONS

-y *dirname* This option allows you to specify a new pathname, *dirname*, for the location of **obj2coff**. The new pathname for **obj2coff** is *dirname/obj2coff*. The default pathname for **obj2coff** is */usr/apollo/bin*.

FILES

<i>/usr/apollo/bin/obj2coff</i>	obj2coff converter
<i>/tmp/obj/*</i>	Temporary files
<i>/tmp/coff/*</i>	Temporary files

SEE ALSO

obj2coff(1), **ar(1)**.

NAME

lcm – load a color map

SYNOPSIS

lcm [**-p** *pathname*]

DESCRIPTION

lcm loads a color map from a file that specifies a set of color map entries. Each entry establishes an association between an index and a color value. When the DM is initially loaded, it sets the node's color map from the file in `'node_data/etc/dm_display/color_map'`.

If no *pathname* is given, **lcm** loads the color map from `'node_data/etc/dm_display/color_map'`. In this case, all 16 colors (that is, color entries for color slots 0-15) are reloaded. If you specify a *pathname*, **lcm** reads the given file and tries to load the colors associated with the indexes.

NOTE

If there are direct mode graphics programs running that have changed the color values for color slots 0-15, the execution changes the colors in these windows as well as resetting the DM's colors.

OPTIONS

- p** *pathname* Specify the file that contains the color values for red, green, and blue. The format of this file should be identical to the DM's color map file, `'node_data/etc/dm_display/color_map'`. For more information about the format of this file, please refer to the manual *Programming with Domain Graphics Primitives*.
- r** *pathname* This option loads the true color map. If no *pathname* is given, **lcm** loads the true color map from the file in `'node_data/etc/dm_display/color_map.rgb'`. This option is currently available only on the DN10000VS. The default color map is a linear ramp that goes from 0 to 255 (0 0 0, 1 1 1, 2 2 2, etc.).
- l** This option lists the contents of the display's color map. The output has the same format as the color map in `'node_data/etc/dm_display/color_map'`. The display's color map is not changed if you use this option.
- s** *i r g b* Sets the color slot *i* to the *rgb* value specified. *i* is a decimal integer, from 0 through 15 on 4-plane displays, and from 0 through 255 on displays with 8 or more planes. *r*, *g*, and *b* are integers ranging from 0 to 255 that represent the brightness of the red, green, and blue components of the color value at slot *i*. You may use more than one "-s" option if you want to set more than one color slot.
- rgb444** This option loads the 12-plane true color map. If the **-p** option is not used, **lcm** loads the 12-plane true color map from the file in `'node_data/etc/dm_display/color_map.rgb444'`. This option is

supported on the Series 10000VS. The default 12-plane true color map is a 16-entry linear ramp. The color values in these slots go from 0 0 0 to 255 255 255 in increments of 17 (0 0 0 in color slot 0, 17 17 17 in color slot 1, 34 34 34 in color slot 2, etc).

EXAMPLES

Load the DM's color map found in the file *'node_data/etc/dm_display/color_map'*.

```
$ lcm
```

Load the color map specified in the file **my_colormap**.

```
$ lcm -p my_colormap
```

Load the true color map.

```
$ lcm -r
```

Set color slot 0 to red.

```
$ lcm -s 0 255 0 0
```

NAME

ld – link editor for common object files

SYNOPSIS

ld [*options*] [**-A** *option*, [*argument*] ...] *file* ...

DESCRIPTION

ld combines several object files into one, performs relocation, resolves external symbols, and supports symbol table information for symbolic debugging. In the simplest case, you specify the names of several object programs, and **ld** combines the objects, producing an object module that can be either executed or, if you specify the **-r** option, used as input for a subsequent **ld** run. The output of **ld** is left in **a.out**. By default this file is executable if no errors occurred during the load. If any input *file* is not an object file, **ld** assumes it is either an archive library or a text file containing link-editor directives.

If any argument is a library, it is searched exactly once, at the point it is encountered in the argument list. **ld** loads only routines that define an unresolved external reference. It searches the library (archive) symbol table sequentially with as many passes as it needs to resolve external references that library members can satisfy (see **ar(5)**). Thus, the ordering of library members is functionally unimportant, unless multiple library members define the same external symbol.

OPTIONS

- d** Force definition of common storage even if the **-r** flag is present.
- e *epsym*** Set the default entry-point address for the output file to be that of the symbol *epsym*.
- f *fill*** Set the default fill pattern for “holes” within an output section as well as initialized *bss* sections. The argument *fill* is a two-byte constant.
- lx** Search the library named *libx.a*. **ld** searches a library when it encounters the library name, so the placement of a **-l** is significant. By default, **ld** searches for libraries in the directories **/lib** and **/usr/lib**. If the **SYSTYPE** environment variable is set to **bsd4.3**, **ld** will also search for libraries in **/usr/local/lib**. You can specify additional library directories with the **-L** option.
- m** Produce a map or listing of the input/output sections on the standard output.
- o *outfile*** Name the final output file *outfile*. By default, *outfile* is **a.out**. If you specify a different name, the system leaves any existing **a.out** files undisturbed.
- r** Retain relocation entries in the output object file. Relocation entries must be saved if the output file is to become an input file in a subsequent **ld** run. The link editor will not complain about unresolved references, and the output file will not be executable unless **-a** is also specified.

- a** Create an absolute file. This is the default if you do not specify **-r**. Used with the **-r** option, **-a** allocates memory for common symbols.
- s** Strip line-number entries and symbol-table information from the output object file. Specifying this option is equivalent to using the **strip(1)** utility and is useful if you want to reduce the size of the object module. Note, however, that removing this information from a program makes it impossible to debug it with a source-level debugger (**dbx** or **dde**).
- T *systype*** Define the target system type (*systype*) for the compiled object.
- t** Trace; print the name of each file as it is processed.
- u *sym*** Enter *sym* as an undefined symbol in the symbol table. This is useful for loading entirely from a library, since the symbol table is initially empty and needs an unresolved reference to force **ld** to load the first routine. The placement of this option on the **ld** line is significant; you must put it before the library that will define the symbol.
- x** Do not preserve local symbols in the output symbol table; enter external and static symbols only. This option saves some space in the output file. This option has no effect on objects compiled on Domain/OS systems, since Domain/OS compilers do not generate local symbol information.
- L*dir*** Changes the algorithm of searching for **libx.a** to look in *dir* before looking in **/lib** and **/usr/lib** (and **/usr/local/lib** for **SYSTYPE = bsd4.3**). This option is useful if you have different versions of a library and you want to specify which one **ld** should use. This option is effective only if it precedes a **-l** option on the command line.
- M** Produce a primitive load map, listing the names of the files that will be loaded.
- V** Output a message giving information about the version of **ld** being used.
- VS *num*** Use *num* as a decimal version stamp identifying the **a.out** file that is produced. The version stamp is stored in the optional header.
- Y [LU], *dir*** Change the default directory used for finding libraries. If you specify **L**, **ld** replaces **/lib** as the first default directory with *dir*. If you specify **U**, and **ld** was built with a second default directory, **/usr/lib**, **ld** replaces that directory with *dir*. If you specify **U**, and **ld** was built with only one default directory, **ld** prints a warning and ignores the option.
- ysym** Indicate each file in which *sym* appears, its type, and whether the file defines or references it. You can specify many such

options to trace many symbols. The placement of `-ysym` is significant; symbol tracing is not done on files that precede the `-y` option.

- `-A` The 4.3BSD version of this option is not supported in Domain/OS BSD. Instead, `-A` identifies Domain/OS BSD-specific options.

The following `ld` options are not supported in Domain/OS BSD: `-D`, `-n`, `-N`, `-S`, `-X`, and `-z`.

Domain/OS BSD EXTENSIONS

The Domain/OS BSD version of `ld` includes support for features that are not available on 4.3BSD. Domain/OS `ld` supports the following extensions: static resource information records (`.sri`), module information records (`.mir`), and control of global variable visibility.

You must precede each of the following arguments with the `-A` option.

- `-A sys[type], sys` Sets the environment variable `SYSTYPE` to `sys` while `ld` is running. If `sys` is `any`, `SYSTYPE` is not reset. A static resource information (SRI) record for `systype` is produced. This option is useful to set the resolution of `systype`-dependent links at run time. For example, if your `systype` is `bsd4.3`, and you specify the following: `-A systype, sys5.3`, the file `/usr/lib/libm.a` resolves to `/sys5.3/usr/lib/libm.a`, instead of `/bsd4.3/usr/lib/libm.a`.
- `-A run[type], sys` This option determines the system call semantics of the object module. For example, if you specify `-A systype, any`, and specify `-A runtype, bsd4.3`, the executable resolves pathnames according to your current `SYSTYPE` value, but always uses `bsd4.3` system-call semantics.
- `-A stacksize, hexnum` Produces a stacksize (SRI) with the specified value; `hexnum` is a one-digit to eight-digit hexadecimal number, optionally preceded by `0x` or `0X`.
- `-A exp[unge], sym1, sym2, ...` Remove the defined global symbol from the symbol table. No subsequent link runs (using `ld` or `bind`) can resolve to this symbol. The symbol is not entered into the KGT (Known Global Table) if this object is installed, nor is it visible if this object is part of an archive.
- `-A allexp[unge] [-A keep[sym], sym1, sym2, ...]` Expunge all defined symbols except those specified. If this option appears multiple times, all symbols specified are kept in the symbol table.
- `-A looks[ection], sec1, sec2, ...` Makes the named section available for sharing with a public section in an installed library.

- A alllooks[ection]** Makes all sections available for sharing with their counterpart public sections in an installed library.
- A marks[ection], *sec1*, *sec2*, ...** Makes the specified section names (*sec1*, *sec2*, ...) public. Affects only those object files that are destined to be installed as an installed library.
- A allmarks[ection]** Makes all sections public. Affects only those object files that are destined to be installed as an installed library.
- A noloooks[ection], *sec1*, *sec2*, ...** Makes the named sections (*sec1*, *sec2*, ...) unavailable for sharing.
- A allnoloooks[ection]** Makes all data named sections unavailable for sharing.
- A nomarks[ection], *sec1*, *sec2*, ...** Makes the named sections private.
- A allnomarks[ection]** Makes all sections private.
- A inlib, *pathname1*, *pathname2*, ...** Specifies the pathnames of the libraries to be installed at load time.
- A noinlib, *pathname1*, *pathname2*, ...** Deletes the named libraries from the list of libraries to be installed at load time.
- A loadhigh** Creates an SRI record to instruct the loader to load the object at the "high" end of memory (for position independent code).
- A module, *name*** Specifies the object module name of the output binary file. This name will be stored in an object file module information record (MIR). The default name of the object module is the name of the first object module read.
- A nosys[tem]** Do not make system globals visible to **ld**. Turn off the check in KGT (Known Global Table) and installed libraries.
- A allres[olved]** Exit with an error status if unresolved references exist. This is the default.
- A noallres[olved]** Terminate successfully even if unresolved references exist (if there are no other errors during linking). This option is useful when you run **ld** from shell scripts or other drivers, such as **/bin/cc**, or **/bin/make**.

BUGS

Through its options and input directives, the common link editor gives users great flexibility; however, those who use the input directives must assume some added responsibilities. Input directives and options should ensure the following properties for programs:

- C defines a zero pointer as null. A pointer to which zero is assigned must not point to any object. To satisfy this, do not place any object at virtual address zero in the program's address space.
- When you call **ld** through **cc(1)**, a startup routine is linked with your program. This routine calls after execution of the main program. If you call **ld** directly, you must ensure that the program always calls **exit(2)** rather than falling through the end of the entry routine.

The symbols **etext**, **edata**, and **end** (see **end(3C)**) are reserved and are defined by **ld**. It is incorrect for a user program to redefine them.

If **ld** does not recognize an input file as an object file or an archive file, it assumes that it contains **ld** directives and attempts to parse it. This occasionally produces an error message complaining about syntax errors.

Arithmetic expressions can have only one forward-referenced symbol per expression.

FILES

/lib/libx.a	Libraries
/usr/lib/libx.a	Libraries
/usr/local/lib/libx.a	Libraries (bsd4.3 only)
a.out	Output file

SEE ALSO

ar(1), **cc(1)**, **end(3)**, **ar(5)**, **a.out(5)**, **systype(1)**

NAME

leave -- remind you when you have to leave

SYNOPSIS

leave [**[+]***hhmm*]

DESCRIPTION

leave waits until the specified time, then reminds you that you have to leave. You are reminded five minutes and one minute before the actual time, at the time, and every minute thereafter. When you log off, **leave** exits just before it would have printed the next message.

Specify the time of day in the form *hhmm* where *hh* represents the time in hours (on a 12 or 24 hour clock) and *mm* represents minutes. **leave** converts all times to a 12 hour clock, and assumes them to be in the next 12 hours.

If you precede the time by a +, the alarm will go off in *hh* hours and *mm* minutes from the current time.

If you do not specify an argument, **leave** prompts with "When do you have to leave?". Typing a newline in reply causes **leave** to exit; otherwise the reply is assumed to be a time. This form is suitable for inclusion in a **.login** or **.profile** file.

leave ignores interrupt, quit, and terminate signals. To get rid of it, you should either log off or use the **kill -9** signal, giving its process ID.

SEE ALSO

calendar(1)

NAME

lex – generator of lexical analysis programs

SYNOPSIS

lex [**-tvfn**] [*file*] ...

DESCRIPTION

lex generates programs to be used in simple lexical analysis of text. The input *files* (standard input is the default) contain regular expressions to be searched for, and actions written in C to be executed when expressions are found.

lex generates a C source program, **lex.yy.c**, to be compiled thus:

```
cc lex.yy.c -ll
```

This program, when run, copies unrecognized portions of the input to the output, and executes the associated C action for each regular expression that is recognized.

OPTIONS

- t** Place the result on the standard output instead of in file **lex.yy.c**.
- v** Print a one-line summary of statistics of the generated analyzer.
- n** The opposite of **-v**; **-n** is the default.
- f** Compile faster: don't bother to pack the resulting tables. This option is limited to small programs.

EXAMPLE

To draw **lex** instructions from the file **scanner.l** and place the output in **lex.yy.c**, execute the following:

```
% lex scanner.l
```

The following text is an example of a **lex** program that would be put into a **lex** command file. This program converts uppercase to lowercase, removes blanks at the end of lines, and replaces multiple blanks by single blanks.

```
%%
[A-Z] putchar(yytext[0]+'a'-'A');
[ ]+$ ;
[ ]+ putchar(' ');
```

SEE ALSO

awk(1), **yacc**(1), **sed**(1)

M. E. Lesk and E. Schmidt, *LEX – Lexical Analyzer Generator*

NAME

lint – a C program verifier

SYNOPSIS

lint [**-abchnpvx**] *file* ...

DESCRIPTION

lint attempts to detect features of C program *files* that are likely to be erroneous, non-portable, or wasteful. It also checks the type usage of the program more strictly than the compilers, searching for unreachable statements, loops not entered at the top, automatic variables declared and not used, and logical expressions whose value is constant. Moreover, **lint** looks for functions that return values in some places and not in others, functions that are called with varying numbers of arguments, and functions whose values are not used.

By default, **lint** assumes that all the *files* are to be loaded together. It checks them for mutual compatibility. Function definitions for certain libraries are available to **lint**. These libraries are referenced by a conventional name, such as **-lm**, in the style of **ld**(1). Arguments ending in **.ln** are also treated as library files.

OPTIONS

You can use any number of the options listed below. **lint** also recognizes the **-D**, **-U**, and **-I** options of **cc**(1) as separate arguments.

- p** Attempt to check portability to the IBM and GCOS dialects of C.
- h** Apply a number of heuristic tests to attempt to intuit bugs, improve style, and reduce waste.
- b** Report **break** statements that cannot be reached. This is not the default, because most **lex**(1) and many **yacc**(1) outputs produce dozens of such comments.
- v** Suppress complaints about unused arguments in functions.
- x** Report variables referred to by external declarations, but never used.
- a** Report assignments of long values to integer variables.
- c** Complain about casts that have questionable portability.
- u** Do not complain about functions and variables used and not defined, or defined and not used. This is suitable for running **lint** on a subset of files from a larger program.
- n** Do not check compatibility against the standard library.
- z** Do not complain about structures that are never defined (for example, allow the use of a structure pointer without knowing its contents).
- C** Create **lint** libraries, as in **lint -Cfoo files . . .** where *files* are the C sources of library **foo**. The result is a file **llib-lfoo.ln** in the correct library format suitable for **linting** programs using **foo**.

C SOURCE COMMENTS

Certain conventional comments in the C source change the behavior of **lint**:

/*NOTREACHED*/

At appropriate points, stops comments about unreachable code.

/*VARARGS*n/**

Suppresses the usual checking for variable numbers of arguments in the following function declaration. Checks the data types of the first *n* arguments. Takes a missing *n* to be 0.

/*NOSTRICT*/

Shuts off strict type checking in the next expression.

/*ARGSUSED*/

Turns on the **-v** option for the next function.

/*LINTLIBRARY*/

At the beginning of a file, shuts off complaints about unused functions in the file.

FILES

/usr/lib/lint/lint[12]	Programs
/usr/lib/lint/lib-1c.ln	Declarations for standard functions
/usr/lib/lint/lib-1c	Human-readable version of the above declarations file
/usr/lib/lint/lib-port.ln	Declarations for portable functions
/usr/lib/lint/lib-port	Human readable . . .
lib-1*.ln	Library created with -C

BUGS

lint does not understand functions that do not return, for example, **exit(2)**.

There are some things you just can't get **lint** to shut up about.

/*NOSTRICT*/ is not implemented in the current version.

SEE ALSO

cc(1), **ld(1)**

NAME

llib — list installed libraries

SYNOPSIS

llib [-a]

DESCRIPTION

The **llib** command lists those libraries which have been installed in the current process via the build-in **llib** shell. These libraries are used to resolve unknown references when loading a program. To find out if a symbol is known and will be used in resolving an unknown reference, use **esa**.

OPTIONS

-a Also list those libraries which are known globally to every process. These libraries are installed at boot time using the configuration information in **/etc/sys.conf**.

SEE ALSO

sh(1), **csh(1)**, **ksh(1)**

NAME

llkob – list locked objects

SYNOPSIS

llkob [*options*]

DESCRIPTION

llkob lists the locked objects resident on volumes mounted on this node, and objects resident in other nodes that are locked by processes running locally.

The listing for each object includes the locking constraints imposed on the object (for example, n-readers XOR 1-writer), the specific lock mode being used (for example, read, write, read-intending-write), the network node ID of the node at which the object is located, the node ID of the node in which the locking process is active, and the name (if it is available) of the object itself.

OPTIONS

- r[emote]** Specify list of only those objects that either reside on this node and are locked by another node, or reside on another node and are locked by this node (that is, those objects whose locks are in some way remote).
- c[ount]** List only a one-line summary of the number of objects locked.

EXAMPLES

\$ llkob

USE	CONSTRAINT	HOME NODE	LOCKING NODE	FILE
W	nR_xor_1W	21	21	/sys/dm/pdb
R	nR_xor_1W	21	21	/sys/dm/fonts/std
W	nR_xor_1W	21	21	--Temporary File--
R	nR_xor_1W	21	21	--Uncataloged Permanent File--
W	nR_xor_1W	21	21	--Display Manager Pad--

\$ llkob -c

locked:102 -- 100 local,2 remote;100 locally locked,2 remotely

NAME

ln - make links

SYNOPSIS

ln [-s] *sourcename* [*targetname*]
ln [-s] *sourcename1* *sourcename2* [*sourcename3* ...] *targetdirectory*

DESCRIPTION

A link is a directory entry referring to a file; the same file (together with its size, all its protection information, and so on) may have several links to it. There are two kinds of links, hard links and symbolic links.

By default **ln** makes hard links. A hard link to a file is indistinguishable from the original directory entry. Any changes to a file are effective independent of the name used to reference the file. Hard links may not span file systems and may not refer to directories.

The -s option causes **ln** to create symbolic links. A symbolic link contains the name of the file it is linked to. The referenced file is used when an **open(2)** operation is performed on the link. A **stat(2)** on a symbolic link returns the linked-to file; you must execute **lstat(2)** to get information about the link. You can use the **readlink(2)** call to read the contents of a symbolic link. Symbolic links may span file systems and may refer to directories.

Given one or two arguments, **ln** creates a link to an existing file *sourcename*. If *targetname* is given, the link has that name; *targetname* can also be a directory in which to place the link; otherwise **ln** places it in the current directory. If only the directory is specified, **ln** makes the link to the last component of *sourcename*.

Given more than two arguments, **ln** makes links in *targetdirectory* to all the named source files. The links made have the same name as the files being linked to.

NOTE

Symbolic links in Domain/OS are implemented as soft links. These are identical in behavior, except that soft links do not have protections associated with the links themselves.

Domain/OS symbolic links are implemented in the parent directory object, not in separate files. Because of this, symbolic links created in the same directory will share the creation/modification information for the most recently created symbolic link.

SEE ALSO

rm(1), **cp(1)**, **mv(1)**, **link(2)**, **readlink(2)**, **stat(2)**, **symlink(2)**

NAME

logger – make entries in the system log

SYNOPSIS

logger [*-t tag*] [*-p pri*] [*-i*] [*-f file*] [*message ...*]

DESCRIPTION

logger provides a program interface to the **syslog(3)** system log module.

You can give **logger** a message on the command line, which is logged immediately, or **logger** can read a file and log each line.

OPTIONS

- t tag* Mark every line in the log with the specified *tag*.
- p pri* Enter the message with the specified priority. You can specify the priority numerically or as a “facility.level” pair. For example, *-p local3.info* logs the message(s) as *informational* level in the *local3* facility. The default is “user.notice.”
- i* Log the process ID of the logger process with each line.
- f file* Log the specified file.
- message* The message to log; if you do not specify one, **logger** logs the *-f* file or standard input.

EXAMPLES

```
% logger System rebooted
```

```
% logger -p local0.notice -t HOSTIDM -f /dev/idmc
```

SEE ALSO

syslog(3), **syslogd(8)**

NAME

login — sign on

SYNOPSIS

login [**-p**] [*username*] [*env-var ...*]

DESCRIPTION

The **login** command signs a user onto the system, and can be subsequently used to change from one user to another. The latter case is the one described here. See *Getting Started with BSD* for information on initially logging in.

If **login** is invoked without an argument, it prompts for a username, and, if appropriate, a password. Echoing is turned off (if possible) while the user types the password, so it will not appear on the written record of the session.

At some installations, you may be prompted for a second “dialup” password by The prompt “dialup password:”. Both passwords are required for a successful login.

After a successful login, accounting files are updated and the operating environment is set from the `~/environ` file if it exists, or from `/etc/environ` if `~/environ` doesn't exist.

If your environment is BSD, you are informed of the existence of mail (see **mail(1)**). For all environments, the message of the day (`/etc/motd`) is printed. Both are suppressed if you have a `.hushlogin` file in your home directory; this is mostly used to make life easier for non-human users, such as **uucp(1C)**.

After setting up the environment, **login** executes a command interpreter (for example, a shell) as specified in the last field of your `/etc/passwd` file entry. If this field in `/etc/passwd` is empty, the default command interpreter is `/bin/sh` for the BSD and SysV environments, and `/com/sh` for the Aegis environment. See **csh(1)**, **ksh(1)**, and **sh(1)** for a description of the shell's startup behavior. Argument 0 of the command interpreter is the name of the command interpreter with a leading dash (“-”).

ENVIRONMENT INITIALIZATION

The **login** command initializes the user and group IDs and the working directory, and modifies the environment as appropriate (see **environ(7)**).

The basic SysV environment is initialized to:

```
HOME=your-log-in-directory
LOGNAME=your-log-in-name
MAIL=/usr/mail/your-log-in-name
NODEID=your-node's-hexadecimal-id
NODETYPE=your-node's-model-number
ORGANIZATION=your-organization-name
PATH=:/bin:/usr/bin:/usr/apollo/bin
PROJECT=your-project-name
SHELL=last-field-of-passwd-entry
SYSTYPE=sys5.3
TERM=your-terminal-type
TZ=timezone-specification
USER=your-log-in-name
```

The basic BSD environment is initialized to:

```
HOME=your-log-in-directory
LOGNAME=your-log-in-name
MAIL=/usr/spool/mail/your-log-in-name
NODEID=your-node's-hexadecimal-id
NODETYPE=your-node's-model-number
ORGANIZATION=your-organization-name
PATH=:/usr/ucb/bin:/usr/bin:/usr/apollo/bin
PROJECT=your-project-name
SHELL=last-field-of-passwd-entry
SYSTYPE=bsd4.3
TERM=your-terminal-type
TZ=timezone-specification
USER=your-log-in-name
```

The `-p` argument causes the remainder of the environment to be preserved, otherwise any previous environment is discarded.

Modifying the Environment

The environment can be expanded or modified by supplying additional arguments to **login**, either at execution time or when **login** requests your log-in name. Arguments can take either the form `xxx` or `xxx=yyy`. Arguments without an equal sign are placed in the environment as

`Ln=xxx`

where *n* is a number starting at 0 and is incremented each time a new variable name is required. Variables containing `=` are placed in the environment without modification. If they already appear in the environment, then they replace the older value, with two exceptions. The variables `PATH` and `SHELL` cannot be changed. Both **login** and **getty** understand simple single-character quoting conventions. Typing a backslash in front of a character quotes it and allows the inclusion of such things as spaces and tabs.

DIAL-UP SECURITY

Sites wishing additional security protection on dial-up lines may want to use these security features, `/etc/d_users` and `/etc/d_passwd`. `/etc/d_users` is simply a file containing a list of users authorized to log in on this node.

`/etc/d_passwd` is a file containing lines of the following format:

```
/bin/sh:encrypted-password
```

where *encrypted-password* is the dial-in password for the specified shell as returned by **crypt**(3). If an entry for the user's log-in shell is not found in this file, the password for `/bin/sh` is used. Sites wishing to restrict root access to specific devices should consult the **ttys**(5) manual page.

NOTES

If the file `/etc/nologin` exists, **login** prints the contents of this file on your terminal and exits. This is used by **shutdown**(8) to stop you from logging in when the system is about to go down.

login is recognized by **sh**(1) and **cs**h(1) and executed directly (without forking).

An undocumented option, **-r** is used by the remote **login** server, **rlogind**(8C) to force **login** to enter into an initial connection protocol. **-h** is used by **telnetd**(8C) and other servers to list the host from which the connection was received.

FILES

/etc/utmp	accounting
/usr/adm/wtmp	accounting
/usr/spool/mail/*	mail
/etc/motd	message of the day
/etc/passwd	password file
/etc/nologin	stops logins
.hushlogin	makes login quieter

DIAGNOSTICS

“Login incorrect,” Username or password cannot be matched.

“no directory,” Either the directory is invalid or the machine where the directory exists can not be reached.

“No shell”, “cannot open password file”, Consult a UNIX system programmer.

BUGS

login(1) is not currently used for logging into the Display Manager or the Server Process Manager, although the procedure used by those programs is similar.

SEE ALSO

mail(1), **passwd**(1), **rlogin**(1c), **sh**(1), **passwd**(5), **environ**(7), **init**(8), **getty**(8), **shutdown**(8), **passwd_override**(5)

NAME

look – find lines in a sorted list

SYNOPSIS

look [**-df**] *string* [*file*]

DESCRIPTION

The **look** command consults a sorted *file* and prints all lines that begin with *string*. It uses binary search.

The options **-d** and **-f** affect comparisons as in **sort**(1). If you don't specify a *file*, **look** assumes **/usr/dict/words** with the collating sequence **-df**.

OPTIONS

-d Use dictionary order. Compare only letters, digits, tabs, and blanks.

-f Fold. Uppercase letters compare equal to lowercase letters.

FILES

/usr/dict/words

SEE ALSO

grep (1), **sort** (1).

NAME

indxbib, **lookbib** – build inverted index for a bibliography, find references in a bibliography

SYNOPSIS

indxbib *database* ...
lookbib [*-n*] *database*

DESCRIPTION

indxbib makes an inverted index from the named *databases* (or files) for use by **lookbib** and **refer**(1). These files contain bibliographic references (or other kinds of information) separated by blank lines.

A bibliographic reference is a set of lines constituting fields of bibliographic information. Each field starts on a line beginning with a percent sign (%) followed by a keyletter, then a blank, and finally the contents of the field, which may continue until the next line starting with a percent sign.

indxbib is a shell script that calls **/usr/lib/refer/mkey** and **/usr/lib/refer/inv**. The first program, **mkey**, truncates words to six characters, and maps uppercase to lowercase. It also discards words shorter than three characters, words among the 100 most common English words, and numbers (dates) less than 1900 or greater than 2000. You can change these parameters; see **refer**(1) for more information. The second program, **inv**, creates an entry file (*.la*), a posting file (*.lb*), and a tag file (*.lc*) in the working directory.

lookbib uses an inverted index made by **indxbib** to find sets of bibliographic references. It reads keywords typed after the greater-than sign (>) prompt on the terminal, and retrieves records containing all these keywords. If no matches occur, **lookbib** simply issues another prompt.

lookbib asks if you need instructions, and prints some brief information if you reply with a word starting with the letter "y".

You can search multiple databases, as long as they have a common index made by **indxbib**. In that case, only the first argument given to **indxbib** is specified to **lookbib**.

If **lookbib** does not find the index files (the *.i[abc]* files), it looks for a reference file with the same name as the argument (minus the suffixes). It creates a file with the suffix *.ig*, suitable for use with **fgrep**. It then uses this file to find references. This method is simpler, but the *.ig* file is slower to use than the *.i[abc]* files, and does not allow the use of multiple reference files.

OPTIONS

-n Do not prompt for instructions.

FILES

database.la, *database.lb*, *database.lc*, where *database* is the first argument, or if this is not present, *database.ig* and *database*.

BUGS

All dates should probably be indexed, because many disciplines refer to literature written in the 1800s or earlier.

SEE ALSO

refer(1), addbib(1), sortbib(1), roffbib(1)

NAME

lorder – find ordering relation for an object library

SYNOPSIS

lorder *file* ...

DESCRIPTION

The input is one or more object or library archive (see **ar(1)**) *files*. The standard output is a list of pairs of object filenames, meaning that the first file of the pair refers to external identifiers defined in the second. You can use **tsort(1)** to process the output, to find an ordering of a library suitable for one-pass access by **ld(1)**.

EXAMPLE

This brash one-liner intends to build a new library from existing **.o** files.

```
% ar cr library `lorder *.o | tsort`
```

NOTE

The need for **lorder** has vanished on Domain/OS systems, since **ar(1)** and **ld(1)** cooperate to create randomly accessed libraries.

FILES

***symref**
***symdef**

BUGS

The names of object files, in and out of libraries, must end with **.o**; nonsense results otherwise.

SEE ALSO

ar(1), **join(1)**, **ld(1)**, **nm(1)**, **sed(1)**, **sort(1)**, **tsort(1)**

NAME

lpq – spool queue examination program

SYNOPSIS

lpq [*+[n]*] [*-l*] [*-Pprinter*] [*job# ...*] [*user ...*]

DESCRIPTION

lpq examines the spooling area used by **lpd**(8) for printing files on the line printer, and reports the status of the specified jobs or of all jobs associated with a user. **lpq** invoked without any arguments reports on any jobs currently in the queue.

For each job submitted (that is, each invocation of **lpr**(1)), **lpq** reports the user's name, the current rank in the queue, the names of files comprising the job, the job identifier (a number which can be supplied to **lprm**(1) for removing a specific job), and the total size in bytes. Job ordering is dependent on the algorithm used to scan the spooling directory and is supposed to be first in, first out (FIFO). Filenames comprising a job may be unavailable (when **lpr**(1) is used as a sink in a pipeline), in which case the file is indicated as “(standard input)”.

If **lpq** warns that there is no daemon present, the **lpc**(8) command can be used to restart the printer daemon.

OPTIONS

- l** Print information about each of the files comprising the job. Normally, **lpq** displays only as much information as fits on one line.
- +[n]** Display the spool queue until it empties. Supplying an optional number immediately after the plus sign (+) indicates that **lpq** should sleep *n* seconds in between scans of the queue.
- Pprinter** Specify a particular printer; otherwise, **lpq** uses the default line printer or the value of the **PRINTER** variable in the environment.
- job#* Look for the print job with the specified number.
- users* Look for the print jobs owned by the specified users.

FILES

- /etc/termcap** For manipulating the screen for repeated display
- /etc/printcap** To determine printer characteristics
- /usr/spool/*** The spooling directory, as determined from **printcap**
- /usr/spool/*/cf*** Control files specifying jobs
- /usr/spool/*/lock** The lock file to obtain the currently active job

SEE ALSO

lpr(1), **lprm**(1), **prf**(1), **lpc**(8), **lpd**(8)

BUGS

Due to the dynamic nature of the information in the spooling directory, **lpq** may report unreliably.

Output formatting is sensitive to the line length of the terminal; this can result in widely spaced columns.

NAME

lpr – print files offline

SYNOPSIS

lpr [*options*] [*file(s)*]

DESCRIPTION

lpr uses a spooling daemon to print the named files when facilities become available. If no names appear, the standard input is assumed. The **-P** option may be used to force output to a specific printer. Normally, **lpr** uses the default printer (site dependent), or the printer specified by the value of the environment variable **PRINTER**.

OPTIONS

The following single letter options are used to notify the line printer spooler that the files are not standard text files. The spooling daemon will use the appropriate filters to print the data accordingly.

- p** Use **pr(1)** to format the files (equivalent to **print**).
- l** Use a filter which allows control characters to be printed and suppresses page breaks.
- t** Assume the files contain data from **troff(1)** (phototypesetter commands).
- n** Assume the files contain data from **ditroff** (device independent troff).
- d** Assume the files contain data in DVI (Stanford) format.
- g** Assume the files contain standard plot data as produced by the **plot(3X)** routines (see also **plot(1G)** for the filters used by the printer spooler).
- v** Assume the files contain a raster image for devices like the Benson Varian.
- c** Assume the files contain data produced by **cifplot(1)**.
- f** Use a filter which interprets the first character of each line as a standard FORTRAN carriage control character.

The remaining single letter options have the following meanings:

- r** Remove the file upon completion of spooling or upon completion of printing (with the **-s** option).
- m** Send mail upon completion.
- h** Suppress the printing of the burst page.
- s** Use symbolic links. Usually files are copied to the spool directory.
- Pprinter**
Force output to print on *printer*.
- C class**
Use *class* as a job classification to print on the burst page.
- J job** Take the following argument (*job*) as the job name to print on the burst page. Normally, the first file's name is used.

- T *title* Make **pr(1)** use *title* as the title instead of the file name.
- #*num* Print multiple copies of output. The *num* argument is the number of copies desired of each file named.
- i[*numcols*] Indent the output. If the next argument is numeric, it is used as the number of blanks to be printed before each line; otherwise, eight characters are printed.
- w*num* Takes *num* as the page width for **pr**.
- s Use **symlink(2)** to link data files rather than trying to copy them so large files can be printed. This means the files should not be modified or removed until they have been printed.
- 1234 *font* Specifies a font to be mounted on font position *i* (*i* equals 1 through 4). The daemon will construct a **.railmag** file referencing **/usr/lib/vfont/name.size**.

EXAMPLES

The command

```
lpr -C EECS foo.c
```

replaces the system name (the name returned by **hostname(1)**) on the burst page by EECS, and prints the file **foo.c**.

The command

```
lpr -#3 foo.c bar.c more.c
```

results in three copies of the file **foo.c**, followed by 3 copies of the file **bar.c**, etc. On the other hand,

```
cat foo.c bar.c more.c | lpr -#3
```

prints three copies of the concatenation of the files.

FILES

/etc/passwd	personal identification
/etc/printcap	printer capabilities data base
/usr/lib/lpd*	line printer daemons
/usr/spool/*	directories used for spooling
/usr/spool/*/cf*	daemon control files
/usr/spool/*/df*	data files specified in "cf" files
/usr/spool/*/tf*	temporary copies of "cf" files

SEE ALSO

lpq(1), **lprm(1)**, **pr(1)**, **prf(1)**, **symlink(2)**, **printcap(5)**, **lpc(8)**, **lpd(8)**

DIAGNOSTICS

If you try to spool too large a file, it will be truncated. **lpr** will object to printing binary files. If a user other than root prints a file and spooling is disabled, **lpr** will print a message saying so and will not put jobs in the queue. If a connection to **lpd** on the local machine cannot be made, **lpr** will say that the daemon cannot be started. Diagnostics may be printed in the daemon's log file regarding missing spool files by **lpd**.

BUGS

Fonts for **troff** and **tex** reside on the host with the printer. It is currently not possible to use local font libraries.

NAME

lprm – remove jobs from the line-printer spooling queue

SYNOPSIS

lprm [*-Pprinter*] [*-*] [*job ...*]

lprm *username ...*

DESCRIPTION

lprm removes jobs from a printer's spool queue. Since the spooling directory is protected from users, **lprm** is normally the only method you can use to remove a job.

lprm without any arguments deletes the currently active job if it is owned by the user who invoked **lprm**. It announces the names of any files it removes and is silent if no jobs in the queue match the request list.

lprm kills off an active daemon, if necessary, before removing any spooling files. If a daemon is killed, a new one is automatically restarted upon completion of file removals.

OPTIONS

<i>-Pprinter</i>	Remove jobs from the queue associated with the specified printer (otherwise the default printer, or the value of the PRINTER variable in the environment is used).
<i>-</i>	Remove all jobs in the queue which you own. If the super-user employs this flag, the spool queue is emptied entirely. The owner is determined by the user's log-in name and host name on the machine where the lpr command was invoked.
<i>job</i>	Remove an individual job, specified by its job number. You can obtain this number from the lpq(1) program.
<i>username</i>	Specifying a username, or a list of usernames, causes lprm to attempt to remove any jobs queued belonging to that user (or users). This form of invoking lprm is useful only to the super-user.

EXAMPLE

The following example illustrates how to find out a file's job number, and remove the file.

```
% lpq -l
1st: ken                               [job #013ucbarpa]
      (standard input)                 100 bytes
% lprm 13
```

FILES

<code>/etc/printcap</code>	Printer characteristics file
<code>/usr/spool/*</code>	Spooling directories
<code>/usr/spool/*/lock</code>	Lock file used to obtain the PID of the current server and the number of the currently active job

DIAGNOSTICS

“Permission denied” if you try to remove files other than your own.

BUGS

Since there are race conditions possible in the update of the lock file, the currently active job may be incorrectly identified.

SEE ALSO

`lpr(1)`, `lpq(1)`, `prf(1)`, `lpd(8)`

NAME

lpctest – generate lineprinter ripple pattern

SYNOPSIS

lpctest [*length* [*count*]]

DESCRIPTION

lpctest writes the traditional “ripple test” pattern on standard output. In 96 lines, this pattern will print all 96 printable ASCII characters in each position. While originally created to test printers, it is quite useful for testing terminals, driving terminal ports for debugging purposes, or any other task where a quick supply of random data is needed.

The *length* argument specifies the output line length if the the default length of 79 is inappropriate.

The *count* argument specifies the number of output lines to be generated if the default count of 200 is inappropriate. Note that if *count* is to be specified, *length* must be also be specified.

NAME

ls – list contents of directory

SYNOPSIS

ls [**-acdfgilqrstu1ACLFRT**] *name* ...

DESCRIPTION

For each directory argument, **ls** lists the contents of the directory. For each file argument, **ls** repeats its name and any other information you request. By default, **ls** sorts the output alphabetically. If you do not specify an argument, **ls** lists the current directory. If you specify several arguments, **ls** sorts the arguments appropriately, but processes file arguments before directories and their contents.

OPTIONS

- l** List in long format, giving mode, number of links, owner, size in bytes, and time of the last modification (see Long List Format, below.) When **ls** lists the sizes of the files in a directory, it prints a total count of blocks, including indirect blocks. If the file is a special file, the size field contains the major and minor device numbers instead of the total number of blocks. If the file is a soft or symbolic link, **ls** prints the pathname of the linked-to file, preceded by “**->**”.
- g** Include the group ownership of the file in a long output.
- t** Sort by the time modified (most recent first) instead of by name.
- a** List all entries. Without this option, **ls** doesn't list entries whose names begin with a period (.).
- s** Give the size, in kilobytes, of each file.
- d** If the argument is a directory, list only its name. This option is often used with **-l** to get the status of a directory.
- L** If the argument is a soft or symbolic link, list the file or directory the link references rather than the link itself.
- r** Reverse the sort order to get reverse alphabetic or oldest first, as appropriate.
- u** Use the time of last access instead of last modification for sorting (with the **-t** option) and/or printing (with the **-l** option).
- c** Use time of last modification of the i-node (file created, mode changed, etc.) for sorting (with the **-t** option) or printing (with the **-l** option).
- i** For each file, print the i-number in the first column of the report.
- 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**. It uses the order in which entries appear in the directory.

- F Mark directories with a trailing slash (/); sockets with a trailing equal sign (=); soft or symbolic links with a trailing @; and executable files with a trailing asterisk (*).
- R Recursively list the subdirectories encountered.
- l Force one-entry-per-line output format. This is the default when output is not to a terminal.
- C Force multicolumn output. This is the default when output is to a terminal.
- q Force printing of non-graphic characters in filenames as the question mark character (?). This is the default when output is to a terminal.
- T If you specify -T with the -l option, it shows the "Domain/OS type" of each file.

LONG LIST FORMAT

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
- l if the entry is a symbolic link
- s if the entry is a socket
- 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 the owner's permissions; the next to permissions to others in the same user-group; the last to all others. Within each set, the three characters indicate permission to read, write, or execute the file as a program, respectively. Execute permission in a directory is interpreted as permission to search the directory. 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.

ls indicates the group-execute permission character as s if the file has the set-group-ID bit set. Likewise, it indicates the user-execute permission character as s if the file has the set-user-ID bit set.

The last character of the mode (normally x or -) is t if the 1000 (octal) bit of the mode is on. See `chmod(1)` for the meaning of this mode.

If an object is protected by an extended access control list (ACL), `ls -l` displays a plus sign (+) following the standard protections. Use the `lsacl(1)` command to list the extended ACL on a file.

EXAMPLES

The following examples show output from `ls -l`

```
% ls -lL /tmp
total 9
-rwxrwxrwx+ 1      brian  12636  Mar 30 15:20 alarm_server.msg_mbx
-rw-rw-rwx+ 1      brian   245    Mar 30 15:18 bar
srw----- 1      brian    0      Mar 30 15:18 df136.cmd
srw----- 1      brian    0      Mar 30 15:18 df136.rpy
-rw-rw-rwx+ 1      brian   128    Mar 30 15:18 foo
-rw-rw-rw- 1      root    144    Mar 30 11:06 llb-faerie
-rw----- 1      brian   483    Mar 30 12:49 mhmaila00546
-rwxrwxr-x 1      brian  1024    Mar 30 12:48 pi_hint
```

The files `alarm_server.msg_mbx`, `bar`, and `foo`, are protected by extended ACLs.

The following output from `ls -l` shows a single object, named `new`.

```
% ls -l
total 97
-rw-rw-r-- 1      brian  3413   Jan 27 10:01 new
```

Reading from left to right, you see that the first character is a dash (-) indicating that `new` is a file. The permissions `rw-rw-r--` indicate that the file's owner and the group have read and write rights (`rw-`), while all others have only read rights (`r--`). The number 1 in the second column indicates the number of links to the file. The remaining columns show that brian is the file's owner, the size is 3413 bytes, it was last modified on January 27, at 10:01, and finally, the filename is `new`.

BUGS

Newline and tab are considered printing characters in filenames.

The output device is assumed to be 80 columns wide.

The option setting based on whether the output is a teletype is undesirable because `ls -s` is much different than `ls -s | lpr`. On the other hand, not doing this setting would make old shell scripts which used `ls` almost certain losers.

FILES

- `/etc/passwd` To get user ID's for `ls -l`.
- `/etc/group` To get group ID's for `ls -g`.

SEE ALSO

`chmod(1)`, `lsacl(1)`, `acl(7)`

NAME

lsacl – list access control list

SYNOPSIS

lsacl [-odfsvmnlaLR] *file* . . .

DESCRIPTION

If you do not specify an *option*, **lsacl** shows the access control list (ACL) associated with the files and directories specified. It lists entries one per line in the following format:

%.%.% prwxk

where %.%.% represents a subject identifier (SID) in person.group.organization form, and **prwxk** represents a set of rights. If one of the letters **prwxk** appears, the associated right is present, if a - appears, it isn't. See **acl(7)** for more information about access rights. If you specify more than one file, the ACL is preceded by the filename.

OPTIONS

- o Lists the ACL associated with the specified objects. This is default if you do not specify an *option*.
- d Shows how new sub-directories created in the specified directory will inherit their protections. (This output is also known as the initial directory ACL.)
- f Shows how new files created in the specified directory will inherit their protections. (This output is also known as the initial file ACL.)
- s Lists any sub-system information. Protected sub-systems are an Aegis analogue to setuid programs, and you should use the commands available in the Aegis environment to manipulate them. This *option* is provided so that UNIX users can be aware of files that use the feature.
- v Selects verbose output. In this mode, each ACL (object, initial file and initial directory) is preceded by a label.
- m Shows the rights mask for extended ACL entries. By default, this information is not shown.
- n Shows the node/network access rights.
- l Shows a long listing, equivalent to selecting all the above *options*.
- a Shows all bits in the rights field, rather than showing [ignore] and [umask]. See **chacl(1)** for a description of these bits.
- L Directs **lsacl** to follow any soft links encountered, and operate on the object to which the link points. Since soft links in Domain/OS do not have ACLs, attempting to change or display the ACL on a link without the -L flag simply results in a warning.
- R Recursively lists the ACLs of any directories specified on the command line.

Only directories have additional fields relating to inheritance of protections for new sub-directories and files. If you specify a non-directory with either the `-d` or `-f` option, `lsacl` ignores them for that object.

SEE ALSO

`chacl(1)`, `cpacl(1)`, `dbacl(1)`, `chmod(1)`, `chgrp(1)`, `chorg(1)`, `ls(1)`, `umask(2)`, `acl(7)`, `chown(8)`, `salacl(8)`

NAME

lty – list installed types

SYNOPSIS

lty [*options*] [*type_name* ...]

DESCRIPTION

lty lists the types currently installed on a volume. It can also be used to list the contents of internal caches for debugging purposes.

type_name (optional) Display information about the specified types. You can use one or more names, and get information about that list of names only. You may not mix type and trait names in the list. If it's a list of trait names, you must also use the **-t** option. If you specify type or trait names, you may also use the **-u**, **-n**, and other options.

OPTIONS

If no options are specified, **lty** lists types installed on the boot volume.

- n node_spec** Specify the node whose type names are to be listed. You may also specify the entry directory of a volume mounted for software installation, as shown in the example below.
- u** Display type UIDs as well as type names.
- glob** Display contents of global type name cache instead of the type file (for debugging only).
- priv** Display the contents of the private (per-user) type name cache instead of the type file (for debugging only).
- t[rait]** Display the list of named traits, instead of the list of types.

EXAMPLES

\$ lty

Local type file

area	bitmap	boot	casehm	ddf	evetype	hdru	ipad
lheap	mbx	mt	nil	null	obj	objlib	pad
pipe	rec	sch	sio	uasc	und		

\$ lty -u -trait

Local trait list

uid	name
26D05884.40003166	io
2717AFC4.B0003166	io_oc
26D058C3.70003166	io_xoc
278096FD.20002ACB	ios_dir

4 entries listed.

In the following example, the disk has been mounted for software installation. The disk's top level directory (cataloged as `/mounted_disk` by the `mount(8)` command) must contain a `sys` directory. If it does not, you get a "types file not found" error.

```
$ /etc/mount/mounted_disk
```

```
$ lty -n /mounted_disk
```

```
Type file for "//my_node/mounted_disk"
```

area	bitmap	boot	casehm	ddf	evetype	hdru	ipad
lheap	mbx	mt	nil	null	obj	objlib	pad
pipe	rec	sch	sio	uasc	und		

SEE ALSO

`crtty(1)`, `dlty(1)`, `intty(1)`, `trtty(1)`, `mount(8)`

NAME

m4 – macro processor

SYNOPSIS

m4 [*file ...*]

DESCRIPTION

m4 is a macro processor intended as a front end for Ratfor, C, and other languages. It processes each *file* in order. If there are no arguments, or if an argument is **-**, **m4** reads the standard input. It writes the processed text on the standard output.

Macro calls have the form

name(*arg1*,*arg2*, . . . , *argn*)

You must place a left parenthesis immediately after the name of the macro if you want to specify arguments. Leading unquoted blanks, tabs, and newlines are ignored while collecting arguments. Potential macro names consist of alphabetic letters, digits, and underscore characters, where the first character is not a digit.

Use left and right single quotation marks (```) to quote strings. The value of a quoted string is the string stripped of the quotes.

When **m4** recognizes a macro name, it collects the macro's arguments by searching for a matching right parenthesis. Macro evaluation proceeds normally during the collection of the arguments, and any commas or right parentheses that appear within the value of a nested call are as effective as those in the original input text. After argument collection, **m4** pushes the value of the macro back into the input stream and scans it again.

BUILT-IN MACROS

m4 uses the built-in macros described below. You can redefine them, but once you do so, the original meaning is lost. Their values are null unless otherwise stated.

define	Install the second argument as the value of the macro whose name is the first argument. Each occurrence of <i>\$n</i> in the replacement text, where <i>n</i> is a digit, is replaced by the <i>n</i> th argument. Argument 0 is the name of the macro; the null string replaces missing arguments.
undefine	Remove the definition of the macro named in its argument.
ifdef	If the first argument is defined, the value is the second argument; otherwise, the value is the third. If there is no third argument, the value is null. The word UNIX is predefined on UNIX versions of m4 .
changequote	Change quote characters to the first and second arguments. changequote without arguments restores the original values (<code>`</code>).
divert	Change the current output stream to its digit-string argument. Discard output diverted to a stream other than 0 through 9. m4 maintains 10 output streams, numbered 0 through 9. The final output is the concatenation of the streams in numerical order; initially, stream 0 is the current stream.

undivert	Produce immediate output of text from diversions named as arguments, or all diversions if no argument is present. You can undivert text into another diversion. Undiverting discards the diverted text.
divnum	Return the value of the current output stream.
dnl	Read and discard characters up to and including the next newline.
ifelse	There are three or more arguments. If the first argument is the same string as the second, then the value in the third argument. If this is not the case, and there are more than four arguments, repeat the process with arguments 4, 5, 6, and 7. Otherwise, the value is either the fourth string (if present) or null.
incr	Return the value of its argument incremented by one. Calculate the value of the argument by interpreting an initial digit-string as a decimal number.
eval	Evaluate the argument as an arithmetic expression, using 32-bit arithmetic. Operators include +, -, *, /, %, and ^ (exponentiation); relationals; and parentheses.
len	Return the number of characters in the argument.
index	Return the position in the first argument where the second argument begins (zero origin), or -1 if the second argument does not occur.
substr	Return a substring of the first argument. The second argument is a zero origin number selecting the first character. The third argument indicates the length of the substring. A missing third argument is assumed to be large enough to extend to the end of the first string.
translit	Transliterate the characters in the first argument from the set given by the second argument to the set given by the third. You cannot use abbreviations.
include	Return the contents of the file named in the argument.
sinclude	Perform the same function as include , but say nothing if the file is inaccessible.
syscmd	Execute the UNIX command given in the first argument. Return no value.
maketemp	Fill in a string of XXXXX in its argument with the current process ID.
errprint	Print the argument on the diagnostic output file.
dumpdef	Print current names and definitions for the named items, or for all items, if you do not specify an argument.

SEE ALSO

`ratfor(1)`;

B. W. Kernighan and D. M. Ritchie, *The M4 Macro Processor*.

NAME

mail – send and receive mail

SYNOPSIS

```
mail [ -v ] [ -i ] [ -n ] [ -s subject ] [ user ... ]
mail [ -v ] [ -i ] [ -n ] -f [ name ]
mail [ -v ] [ -i ] [ -n ] -u user
```

INTRODUCTION

mail is a intelligent mail-processing system, similar in command syntax to **ed**(1), with lines replaced by messages. Note that information about this command refers to **/usr/ucb/mail**; do not confuse it with **/bin/mail**, which is documented under the **bin-mail**(1) command.

To send a message to one or more people, invoke **mail** with arguments that are the names of intended recipients. Then type in your message, followed by an EOF (CTRL/D) at the beginning of a line. Use **-s** on the command line, to specify a subject.

In normal usage **mail** is given no arguments and checks your mail out of the post office, then prints a one-line header of each message there. The current message is initially the first message (numbered 1) and you can use the **print** command (abbreviated as **p**) to print it. You can move among the messages much as you move between lines in **ed**(1), with the commands **-** and **+** moving backwards and forwards. You can also specify absolute message numbers.

After examining a message, you can delete (**d**) the message or reply (**r**) to it. Deletion causes the **mail** program to forget about the message. This is not irreversible; you can undelete (**u**) the message by giving its number, or you can abort the **mail** session by giving the exit (**x**) command.

You can give a list of message numbers to commands such as **print** and **delete**, as arguments to apply to a number of messages at once. Thus **delete 1 2** deletes messages 1 and 2, while **delete 1-5** deletes messages 1 through 5. The asterisk addresses all messages, and (**\$**) addresses the last message; thus you can use the command **top** (which prints the first few lines of a message) in the form **top *** to print the first few lines of all messages.

Use the **reply** command to respond to the originator of a message. The text you then type in, up to an end-of-file, comprises the contents of the message. While you are composing a message, **mail** treats lines beginning with a tilde (**~**) specially. For instance, typing **~m** alone on a line places a copy of the current message into the response, right-shifting it by a tabstop. Other escapes set up subject fields, add and delete recipients to the message, and allow you to escape to an editor to revise the message or to a shell to run some commands.

Use the **quit** (**q**) command to end a **mail** session. **mail** sends messages that you examined to your **mbox** file unless you delete them, in which case it discards them. It sends unexamined messages back to the post office.

You can also create a personal distribution lists so that, for instance, you can send mail to "cohorts" and have it go to a predefined group of people. Define such a list

by placing a line like

```
alias cohorts bill larry shemp
```

in the file **.mailrc** in your home directory. Use the **alias** (**a**) command in **mail**. to display the current list of aliases. You can create system-wide distribution lists by editing **/usr/lib/aliases**. Refer to **aliases(5)** and **sendmail(8)**; these aliases are kept in a different syntax. **mail** expands personal aliases in mail sent to others so that they can **reply** to the recipients. **mail** does not expand system-wide aliases when it sends mail, but any reply returned to the machine has the system-wide alias expanded as all mail goes through **sendmail(8)**.

To send network mail (ARPA, UUCP, Berknet), see **mailaddr(7)** for a description of network addresses.

COMMANDS

Put each command on a line by itself, with arguments following the command word. You can use the full command name or its valid abbreviation. If a command normally takes a message list as an argument but you do not specify a message list, **mail** uses the next message forward that satisfies the command's requirements. If there are no messages forward of the current message, **mail** searches backwards, and if there are no good messages at all, it responds with "No applicable messages" and aborts the command.

- Go to the previous message and print it. If you specify a numeric argument *n*, go to the *n*th previous message and print it.
- ? Print a brief summary of commands.
- ! Execute the shell command that follows.
- Print (P)** This command is like **print** but it also prints out ignored header fields. See also **print**, **ignore**, and **retain**.
- Reply (R)** Reply to originator. This command does not reply to other recipients of the original message.
- Type (T)** Identical to the **Print** command.
- alias (a)** If you do not specify an argument, print all currently-defined aliases. If you specify one argument, print that alias. If you specify more than one argument, create a new alias or change an old one. The **alias** command is usually used in the **~/.mailrc** file.
- alternates (alt)** Inform **mail** that the listed addresses are really you. When you **reply** to messages, **mail** does not send a copy of the message to any address listed on the alternates list. If you specify the **alternates** command with no argument, it displays the current set of alternate names. The **alternates** command is useful if you have several accounts, and is usually used in the **~/.mailrc** file.
- chdir (c)** Change your working directory to the one specified. If you do not specify a directory, change to your log-in directory.

copy (co)	Perform the same action as save , but do not mark the given messages for deletion.
delete (d)	Take a list of messages as argument and mark them for deletion. mail does not save deleted messages in mbox , and they are not available to most other commands.
dp (or dt)	Delete the current message and print the next message. If there is no next message, mail displays "at EOF."
edit (e)	Edit the specified message(s) using the program indicated by the EDITOR string option.
exit (ex or x)	Return immediately to the shell without modifying your system mailbox, your mbox file, or your edit file if using the option -f .
file (fi)	This command is the same as folder .
folders	List the names of the folders in your folder directory.
folder (fo)	Switch to a new mail file or folder. If you do not specify an argument, identify which file you are currently reading. If you specify an argument, write out changes (such as deletions) made in the current file and read in the new file. mail recognizes some special conventions for the name: a pound sign (#) means the previous file, a percent sign (%) means your system mailbox, a percent sign and a username (%user) means user's system mailbox, an ampersand (&) means your ~/mbox file, and +folder means a file in your folder directory (determined by the value of the folder option discussed below).
from (f)	Print the message headers of the list of messages.
headers (h)	Display one windowful of header lines from the messages around the current message. See also the z command.
help	This command is a synonym for ?
hold (ho)	Take a message list and mark each message in it to be saved in the user's system mailbox instead of in mbox . Do not override the delete command.
ignore	Add the list of header fields named to the ignore list. Header fields in the ignore list are not displayed on your terminal when you print a message. This command is very handy for suppressing certain machine-generated header fields. You can use the type and print commands to print a message in its entirety, including ignored fields. If you specify ignore with no arguments, it lists the current set of ignored fields. Note that ignore has been superseded by retain .
mail (m)	Send mail to the specified users and/or distribution groups given as arguments.
mbox	Indicate a list of messages be added to mbox in your home directory when you quit. This is the default action for messages you have read if you have not set the hold option.

- next (n)** Go to the next message in sequence and type it. You can also use a plus sign (+) or a carriage return. If you specify an argument list, **next** displays the next matching message.
- preserve (pre)** This command is a synonym for **hold**.
- print (p)** Take a message list and display each message on your terminal.
- quit (q)** Terminate the session, saving all undeleted, unsaved messages in your **~/mbox** file, preserving all unread messages and those marked with **hold** or **preserve** in your system mailbox, and deleting all other messages. If new mail arrives during the session, display the message "You have new mail". If you specify **quit** while editing a mailbox file with the **-f** flag, save the file with changes. Return to the shell unless the rewrite of the edit file fails, in which case you can escape with the **exit** command.
- reply (r)** Send mail to the sender and all original recipients of the specified message. Reply to current message if you do not specify one. You cannot reply to a message marked for deletion.
- respond** This command is a synonym for **reply**.
- retain** Add the named list of header fields to the "retain list." **mail** displays only the header fields in the retain list when you print a message; it suppresses all other header fields. You can use the **type** and commands to print a message in its entirety. If you specify **retain** with no arguments, it lists the current set of retained fields.
- save (s)** Take a message list and a filename and append each message in turn to the end of the file. **mail** echoes the filename in quotation marks, followed by the line count and character count after the append, on your terminal. It also marks each message for deletion.
- set (se)** If you do not specify an argument, print all option variable values. Otherwise, set an option. Arguments are of the form "*option=value*" (no space before or after =) or, in the case of Boolean variables, "*option*."
- shell (sh)** Invoke an interactive version of the shell.
- size** Print the size in characters of each message in the specified list.
- source (so)** Read **mail** commands from a file.
- top** Print the top few lines of each message in the given list. The variable *toplines* controls the number of lines printed. (The default value is five.)
- type (t)** This command is a synonym for **print**.
- unalias** Take a list of names defined by **alias** commands and discard the remembered groups of users. The group names no longer have any significance.

- undelete (u)** Mark each message in the given list as not deleted.
- unread (U)** Mark each message in the given list as unread. Unread messages return to your system mailbox when you **quit** the session.
- unset** Discard the values of a list of option names; do the inverse of **set**.
- visual (v)** Invoke the display editor (defined by the option string **VISUAL**) on each message in the given list.
- write (w)** This command is similar to **save**, except that only the message body (without the header) is saved. **write** is very useful for tasks such as sending and receiving source-program text over the message system.
- xit (x)** This command is a synonym for **exit**.
- z** Present message headers in windowsful as described in the **headers** command. This command differs from **headers** in that **z** displays the range following that presented by the last **h** or **z** command; thus sequential **z** commands advance through the available messages, increasing the current message number accordingly. Use **z-** to move backwards.

TILDE ESCAPES

When composing messages with **mail**, you can use tilde (~) escape commands to perform special functions. Tilde escapes are recognized only at the beginning of lines. The name "tilde escape" is somewhat of a misnomer because you can use the option **escape** to set the actual escape character.

- ~!command** Execute the indicated shell command, then return to editing the message.
- ~b name ...** Add the *names* to the list of carbon copy recipients but do not show the names in the cc: line (that is, make "blind" carbon copies).
- ~cname...** Add the *names* to the list of carbon copy recipients.
- ~d** Read the file **dead.letter** from your home directory into the message.
- ~e** Invoke the text editor (as specified with the **EDITOR** string option) on the messages collected so far. When the editing session is finished, you can continue appending text to the message.
- ~f messages** Read the *messages* into the message being sent. If you do not specify a *message*, read in the current message.
- ~h** Edit the message header fields by displaying each one in turn and allowing the user to append text to the end or modify the field by using the current terminal erase and kill characters. If **mail** is being run in a pad, you will be prompted for changes to the existing header values. A carriage return will leave the value unchanged, entering a blank in the first position will remove the entry, and entering a new header value will replace the existing header value.
- ~m messages** Read the *messages* into the message being sent, shifted right one tab. If you do not specify a *message*, read the current message.

~p	Print the message collected so far, prefaced by the message header fields.
~q	Abort the message being sent, copying the message to dead.letter in your home directory if save is set.
~r filename	Read the named file into the message.
~s string	Make <i>string</i> the current subject field.
~t name ...	Add <i>names</i> to the direct-recipient list.
~v	Invoke an alternate editor (defined by the VISUAL option) on the messages collected so far. The alternate editor is usually a screen editor. After you quit the editor, you can resume adding text to the end of your message.
~w filename	Write the message into the named file.
~ command	Pipe the message through <i>command</i> as a filter. If the command gives no output or terminates abnormally, retain the original text of the message. The fmt(1) command is often used to rejustify the message.
--string	Insert the <i>string</i> in the message, prefaced by a single tilde (~). If you have changed the escape character, you should double that character in order to send it.

OPTIONS CONTROLLED BY SET AND UNSET

Use the **set** and **unset** commands to control options. Options may be binary, in which case you need only see whether they are set or not, or string, in which case the actual value is of interest. The binary options include the following:

append	Cause messages saved in mbox to be appended rather than prefixed.
ask	Cause mail to prompt you for the subject of each message you send. If you respond with a newline, mail does not send a subject field.
askcc	Cause mail to prompt you for additional carbon copy recipients at the end of each message. Responding with a newline indicates your satisfaction with the current list.
autoprint	Causes the delete command to behave like dp : when you delete a message, mail displays the next one automatically.
debug	The effect is the same as specifying -d on the command line: cause mail to output information useful for debugging mail .
dot	Cause mail to interpret a period alone on a line as the terminator of the message being sent.
hold	Hold read messages in the system mailbox by default.
ignore	Cause interrupt signals from your terminal to be ignored and echoed as @ 's.

ignoreeof	Cause mail to refuse to accept a CTRL/D as the end of a message. This option also applies to mail command mode.
metoo	Include the sender in the expanded group. Usually, when a group containing the sender is expanded, the sender is removed from the expansion.
nosave	Prevent the normal copying of a partial letter to the file dead.letter in your home directory when you abort a message with two interrupts.
Replyall	Reverse the sense of reply and Reply commands.
quiet	Suppress the printing of the version when first invoked.
verbose	Has the same effect as using the -v flag on the command line. When mail runs in verbose mode, the actual delivery of messages is displayed on your terminal.

The following options have string values:

EDITOR	Pathname of the text editor to use in the edit command and ~e escape. If you do not define an editor, mail uses a default.
PAGER	Pathname of the program to use in the more command or when the crt variable is set. If you do not define a paginator, mail uses a default.
SHELL	Pathname of the shell to use in the ! command and the ~! escape. If you do not define a shell, mail uses a default.
VISUAL	Pathname of the text editor to use in the visual command and ~v escape.
crt	Determines how long a message must be before PAGER is used to read it.
escape	If defined, the first character of this option gives the character to use in place of the tilde (~) to denote escapes.
folder	The name of the directory to use for storing folders of messages. If this name begins with a slash (/), mail considers it to be an absolute pathname; otherwise, mail finds the folder directory relative to your home directory.
record	If defined, gives the pathname of the file used to record all outgoing mail. If not defined, outgoing mail is not saved.
toplines	If defined, gives the number of lines of a message to be printed out with the top command. Normally the first five lines are printed.

COMMAND LINE OPTIONS

-v	Set verbose mode: display details of delivery on your terminal.
-i	Ignore tty interrupt signals. This is particularly useful when using mail on noisy phone lines.

- n** Inhibit the reading of `/usr/lib/Mail.rc`.
- s** Specify a subject line when sending mail. Only the first argument after the **-s** option is used as a subject. Make sure to enclose subjects containing spaces in quotation marks.
- f** Read in the contents of your **mbox** (or the specified file) for processing. Upon termination of the session, write undeleted messages back to this file.
- u user** Read in the contents of `/usr/spool/mail/user` for processing. Upon termination of the session, write undeleted messages back to this file.

FILES

<code>/usr/spool/mail/*</code>	Post office
<code>~/mbox</code>	Your old mail
<code>~/mailrc</code>	File giving initial mail commands
<code>/tmp/R#</code>	Temporary for editor escape
<code>/usr/lib/Mail.help*</code>	Help files
<code>/usr/lib/Mail.rc</code>	System initialization file
<code>Message*</code>	Temporary for editing messages

BUGS

Many flags that are not useful to the general user are not documented here.
 Usually, **mail** is just a link to **Mail**, which can be confusing.

SEE ALSO

`binmail(1)`, `fmt(1)`, `newaliases(1)`, `aliases(5)`, `mailaddr(7)`, `sendmail(8)`

NAME

make – update and regenerate programs or groups of programs

SYNOPSIS

make [-f *makefile*] [-p] [-l] [-k] [-s] [-r] [-n] [-b] [-e] [-t] [-q] [*names*]

DESCRIPTION

make allows the programmer to update and regenerate programs and groups of programs through the use of assembled rules and dependencies present in the specified *makefile*.

make executes commands in *makefile* to update one or more target *names*. *Name* is typically a program. If no -f option is present, *makefile*, *Makefile*, and the Source Code Control System (SCCS) files, *s.makefile* and *s.Makefile*, are tried in order. If you use a dash (-) in place of *makefile*, the standard input is taken. More than one -*makefile* argument pair may appear.

make updates a target when its dependents have been more recently created or modified. All prerequisite files of a target are added recursively to the list of targets. Missing files are considered to be out-of-date.

Makefile contains a sequence of entries that specify dependencies. The first line of an entry is a blank-separated, non-null list of targets, followed by a colon (:), and then a (possibly null) list of prerequisite files or dependencies. If a name appears on the left of more than one single-colon line, it is dependent on all the names to the right of the colon on those lines, though only one command sequence may be specified. If a name appears on a line with a double colon (::), the command sequence following that line is performed if the name is out of date with the names to the right of the double colon. This command sequence is not affected by other double colon lines on which that name may appear. Text following a semicolon (;), as well as all following lines beginning with a tab, are shell commands to be executed in updating the target. The first non-empty line beginning with a character other than tab or pound sign (#) will begin a new dependency or macro definition. Shell commands may be continued across lines with a <backslash><newline> sequence. Everything printed by **make** (except the initial tab) is passed directly to the shell as it is present in the *makefile*. For these reasons,

```

    echo a\
    b
produces
    ab

```

in the same manner as would occur in the shell.

The pound sign (#) and newline surround comments.

The following *makefile* establishes a dependency where **pgm** depends on two files **a.o** and **b.o**, and they in turn depend on their corresponding source files (**a.c** and **b.c**) as well as a common file **incl.h**:


```

pgm: a.o b.o
      cc a.o b.o -o pgm
a.o: incl.h a.c
      cc -c a.c
b.o: incl.h b.c
      cc -c b.c

```

Command lines are executed one at a time, each by its own shell. In the makefile, the **make** macro **SHELL** can be used to set the shell for execution. The default shell for execution is **/bin/sh**. The following special characters may be used to begin a shell command singly, or in combination, to specify the appropriate behavior(s).

A line is printed when it is executed unless the **-s** option is present, or the entry **.SILENT:** is in *makefile*, or unless the initial character sequence contains a **@**. The **-n** option specifies printing without execution; however, if the shell command line contains the string **\$(MAKE)** or begins with the **+** special character, the line is always executed (see discussion of the **MFLAGS** macro under *Environment*). The **-t** (touch) option updates the modified date of a file without executing any commands.

Commands returning non-zero status normally terminate **make**. If the **-i** option is present, or the entry **.IGNORE:** appears in *makefile*, or the initial character sequence of the command contains **-**, the error is ignored. If the **-k** option is present, work is abandoned on the current entry, but continues on other branches that do not depend on that entry.

The **-b** option allows old makefiles (those written for the old version of **make**) to run without errors.

Interrupt and quit cause the target to be deleted unless the target is a dependent of the special name **.PRECIOUS**.

OPTIONS

- f *makefile*** Description file name. *makefile* is assumed to be the name of a description file. A filename of **-** denotes the standard input. If there are no **-f** arguments, the file named **makefile** or **Makefile** or **s.[mM]akefile** in the current directory is read. The contents of the description files override the built-in rules if they are present.
- p** Prints out the complete set of macro definitions and target descriptions.
- i** Ignores error codes returned by invoked commands. This mode is entered if the fake target name **.IGNORE** appears in the description file.
- k** Abandons work on the current entry if it fails, but continues on other branches that do not depend on that entry.
- s** Silent mode. Does not print command lines before executing. This mode is also entered if the fake target name **.SILENT** appears in the description file.
- r** Does not use the built-in rules.

- n** No execute mode. Prints commands, but does not execute them. Even lines beginning with an **@** are printed.
- b** Compatibility mode for old makefiles.
- e** Environment variables override assignments within makefiles.
- t** Touches the target files (causing them to be up-to-date) rather than issue the usual commands.
- q** Question. The **make** command returns a zero or non-zero status code depending on whether the target file is or is not up-to-date.
- d** Prints **make** internal debugging information.
- P** Portable. **make** reads the environment, assuming all variables to be macro definitions and processing them as such. The environment variables are processed before any makefile and after the internal rules. As such, macro assignments in a makefile override environment variables.

Note: The **-p**, **-b**, and **-e** options are not normally found in BSD environments, but have been added for compatibility with SysV.

SPECIAL NAMES

- .DEFAULT** If a file must be made but there are no explicit commands or relevant built-in rules, the commands associated with the name **.DEFAULT** are used if it exists.
- .PRECIOUS** Dependents of this target will not be removed when quit or interrupt are hit.
- .SILENT** Same effect as the **-s** option.
- .IGNORE** Same effect as the **-i** option.

ENVIRONMENT

The **-e** option causes the environment to override the macro assignments in a makefile. Suffixes and their associated rules in the makefile will override any identical suffixes in the built-in rules.

MFLAGS always contains the current input options, including the leading **-**. This proves very useful for "super-makes." In fact, as noted above, when the **-n** option is used, the command **\$(MAKE)** is executed anyway; hence, one can perform a **make -n** recursively on a whole software system to see what would have been executed. This is because the **-n** is put in **MFLAGS** and passed to further invocations of **\$(MAKE)**. This is one way of debugging all of the makefiles for a software project without actually doing anything.

make also supports the **MAKEFLAGS** macro. This is for compatibility with SysV environments.

INCLUDE FILES

If the string **include** appears as the first seven letters of a line in a *makefile*, and is followed by a blank or a tab, the rest of the line is assumed to be a file name and will be read by the current invocation, after substituting for any macros.

MACROS

Entries of the form *string1* = *string2* are macro definitions. *String2* is defined as all characters up to a comment character or an unescaped newline. Subsequent appearances of $\$(string1[:subst1=[subst2]])$ are replaced by *string2*. The parentheses are optional if a single character macro name is used and there is no substitute sequence. The optional *:subst1=subst2* is a substitute sequence. If it is specified, all non-overlapping occurrences of *subst1* in the named macro are replaced by *subst2*. Strings (for the purposes of this type of substitution) are delimited by blanks, tabs, newline characters, and beginnings of lines. An example of the use of the substitute sequence is shown under *Libraries*.

INTERNAL MACROS

There are five internally maintained macros which are useful for writing rules for building targets.

- \$*** The macro **\$*** stands for the file name part of the current dependent with the suffix deleted. It is evaluated only for inference rules.
- \$@** The **\$@** macro stands for the full target name of the current target. It is evaluated only for explicitly named dependencies.
- \$<** The **\$<** macro is only evaluated for inference rules or the **.DEFAULT** rule. It is the module which is out-of-date with respect to the target (i.e., the “manufactured” dependent file name). Thus, in the **.c.o** rule, the **\$<** macro would evaluate to the **.c** file. An example for making optimized **.o** files from **.c** files is:

```
.c.o:
    cc -c -O $*.c
```

or:

```
.c.o:
    cc -c -O $<
```

- \$?** The **\$?** macro is evaluated when explicit rules from the makefile are evaluated. It is the list of prerequisites that are out-of-date with respect to the target; essentially, those modules which must be rebuilt.
- \$%** The **\$%** macro is only evaluated when the target is an archive library member of the form *lib(file.o)*. In this case, **\$@** evaluates to *lib* and **\$%** evaluates to the library member, *file.o*.

Four of the five macros can have alternative forms. When an upper-case **D** or **F** is appended to any of the four macros, the meaning is changed to “directory part” for **D** and “file part” for **F**. Thus, **\$(@D)** refers to the directory part of the string **\$@**. If there is no directory part, **/** is generated. The only macro excluded from this alternative form is **\$?**.

VPATH is another special macro. The **VPATH** macro is a list of directories separated by colons. When **make** searches for a file as a result of a dependent relation, it first searches the current directory, followed by each of the directories on the **VPATH** list. If it finds the file, it uses the actual path to the file, as opposed to just the filename. If **VPATH** is not defined, **make** searches only the current directory.

VPATH is useful when compiling several programs from the same source. The source may be contained in one directory while each set of object files, with a separate makefile, may reside in individual subdirectories. The **VPATH** macro would point to the source directory in this case.

SCCSDIR is another special macro. The **SCCSDIR** macro, when specified as **SCCS**, allows **make** to find **s.** files in the sub-directory **SCCS**. Otherwise, **make** will only find **s.** files in the current directory.

SUFFIXES

Certain names (for instance, those ending with **.o**) have inferable prerequisites such as **.c**, **.s**, etc. If no update commands for such a file appear in *makefile*, and if an inferable prerequisite exists, that prerequisite is compiled to make the target. In this case, **make** has inference rules that allow files to be built from other files by examining the suffixes and determining an appropriate inference rule to use.

A tilde in the rules refers to an **SCCS** file. See **sccsfile(4)** for more information about these files. Thus, the rule **.c~.o** transforms an **SCCS C** source file into an object file (**.o**). Because the **s.** of the **SCCS** files is a prefix, it is incompatible with **make**'s suffix point of view. Hence, the tilde is a way of changing any file reference into an **SCCS** file reference.

make infers prerequisites for files for which *makefile* gives no construction commands. Prerequisites are inferred according to selected suffixes listed as the *prerequisites* for the special name **.SUFFIXES**. Multiple lists accumulate, while an empty list clears the information that came before. Order is significant. The first possible name for existent files and rules is inferred as described in the next paragraph. The default list is defined in **/usr/lib/make/standard**.

A rule with only one suffix (i.e., **.c**) is the definition of how to build **x** from **x.c**. In effect, the other suffix is null. This is useful for building targets from only one source file (e.g., shell procedures, simple C programs).

The **-p** option for printing the internal rules will display the list of suffixes implemented on the current machine. Multiple suffix lists accumulate; **.SUFFIXES**: with no dependencies clears the list of suffixes.

INFERENCE RULES

The first example can be done more briefly.

```
pgm: a.o b.o
    cc a.o b.o -o pgm
a.o b.o: incl.h
```

This is because **make** has a set of internal rules for building files. The user may add rules to this list by simply putting them in the *makefile*.

Certain macros are used by the default inference rules to permit the inclusion of optional matter in any resulting commands. For example, **CFLAGS**, **LFLAGS**, and **YFLAGS** are used for compiler options to **cc(1)**, **lex(1)**, and **yacc(1)**, respectively. Again, the previous method for examining the current rules is recommended.

The inference of prerequisites can be controlled. The rule to create a file with suffix `.o` from a file with suffix `.c` is specified as an entry with `.c.o:` as the target and no dependents. Shell commands associated with the target define the rule for making a `.o` file from a `.c` file. Any target that has no slashes in it and starts with a dot is identified as a rule and not a true target.

LIBRARIES

If a target or dependency name contains parentheses, it is assumed to be an archive library, the string within parentheses referring to a member within the library. Thus `lib(file.o)` and `$(LIB)(file.o)` both refer to an archive library which contains `file.o`. (This assumes the `LIB` macro has been previously defined.) The expression `$(LIB)(file1.o file2.o)` is not legal.

Rules pertaining to archive libraries have the form `XX.a` where the `XX` is the suffix from which the archive member is to be made. An unfortunate byproduct of the current implementation requires the `XX` to be different from the suffix of the archive member. Thus, one cannot have `lib(file.o)` depend upon `file.o` explicitly. The most common use of the archive interface follows. Here, we assume the source files are all C type source:

```
lib:  lib(file1.o) lib(file2.o) lib(file3.o)
      @echo lib is now up-to-date

.c.a:
      $(CC) -c $(CFLAGS) $<
      $(AR) $(ARFLAGS) $@ $*.o
      rm -f $*.o
```

In fact, the `.c.a` rule listed above is built into `make` and is unnecessary in this example. A more interesting, but more limited example of an archive library maintenance construction follows:

```
lib:  lib(file1.o) lib(file2.o) lib(file3.o)
      $(CC) -c $(CFLAGS) $(?:.o=.c)
      $(AR) $(ARFLAGS) lib $?
      rm $?
      @echo lib is now up-to-date

.c.a:;
```

Here the substitution mode of the macro expansions is used. The `$?` list is defined to be the set of object file names (inside `lib`) whose C source files are out-of-date. The substitution mode translates the `.o` to `.c`. (Unfortunately, one cannot as yet transform to `.c`; however, this may become possible in the future.) Note also, the disabling of the `.c.a:` rule, which would have created each object file, one by one. This particular construct speeds up archive library maintenance considerably. This type of construct becomes very cumbersome if the archive library contains a mix of assembly programs and C programs.

NOTES

Some commands return non-zero status inappropriately; use `-i` to overcome the difficulty.

BUGS

Filenames with the equal sign (=), colon (:), or at sign (@) characters will not work. Commands that are directly executed by the shell, notably `cd(1)`, are ineffectual across newlines in **make**. The syntax (*lib(file1.o file2.o file3.o)*) is illegal. You cannot build *lib(file.o)* from *file.o*. The macro `$(a:.o=.c)` does not work. Named pipes are not handled well.

FILES

`[Mm]akefile`, `s.[Mm]akefile`, and `$(SCCSDIR)/s.[Mm]akefile`
`/bin/sh`

SEE ALSO

`cc(1)`, `lex(1)`, `yacc(1)`, `printf(3S)`, `cd(1)`, `sh(1)`.

NAME

man – display reference manual information

SYNOPSIS

man [-] [-**M** *path*] [*section*] *title* ...
man -**k** *keyword* ...
man -**f** *file* ...

DESCRIPTION

man displays information from the *BSD Command Reference* and *BSD Programmer's Reference*. **man** displays entries in an edit pad (if you are using a Domain workstation) or a terminal screen. It allows you to obtain one-line descriptions of commands specified by name, or for all commands whose description contains any of a set of keywords. It can also provide on-line access to the sections of the printed manual.

When the standard output is connected to a Display Manager pad (the normal case when you are using a Domain workstation), **man** displays the manual page in a read-only edit pad. You can resize the pad, scroll through or copy portions of the text, turn the pad into an icon, or do any other operation allowed on a read-only edit pad. When you finish reading the manual entry, close the window by positioning the cursor in the window and pressing <EXIT>.

man looks for the manual page in either of two forms, the **nroff** source or preformatted pages. If either version is available, the manual page is displayed. If the preformatted version is available, and it has a more recent modify time than the **nroff** source, it is displayed. Otherwise, the manual page is usually formatted with **nroff** and displayed.

NOTE: Domain/OS provides only the preformatted manual pages.

OPTIONS

- | | |
|------------------------|---|
| <i>section</i> | Look for title in the given <i>section</i> . The <i>section</i> argument is either an Arabic section number (for example, 3) optionally followed by a single-letter classifier (for instance, 1g, indicating a graphics program in section 1), or one of the words "new," "local," "old," "public," or "apollo." If you omit the <i>section</i> argument, man searches all sections of the manual, giving preference to commands over subroutines in system libraries, and printing the first entry it finds. |
| - | Pipe the output through more (1) with the option -s to crush out useless blank lines and to stop after each page on the screen. Press the spacebar to continue, or CTRL/D to scroll 11 more lines when the output stops. |
| - M <i>path</i> | Search for manual subdirectories in <i>path</i> . Normally man checks in a standard location (/usr/man) for manual information. The search path is a list of directories (separated by colons) in which manual subdirectories may be found, e.g., "/usr/local:/usr/man" . If you supply a search path with the - k or - f options, it must be specified first. |

- k keyword ...** Print a one-line synopsis of each entry whose listing in the table of contents contains the specified *keyword(s)*.
- f file ...** Print a one-line synopsis of each entry that is related to the named system *file*.
- local** Print a local entry. You can add your own entries to the directory */usr/man/cat1* (cat-ell, not cat-one). Filenames in this directory must have the suffix *.1* (a dot followed by a lowercase L).

EXAMPLES

To find out if there are any entries pertaining to the file */etc/rc*:

```
% man -f/etc/rc
rc (8)                - boot time shell script
```

To see a local entry in the file */usr/man/man1/roff.1*:

```
% man local roff
```

NOTES

Since Domain/OS does not supply nroff sources for BSD manual pages, you cannot print copies on local printers.

BUGS

The manual is supposed to be reproducible either on the phototypesetter or on a typewriter. However, on a typewriter some information is necessarily lost.

FILES

<i>/usr/man</i>	Standard manual area
<i>/usr/man/man?/*</i>	Directories containing source for manuals (not supplied on Domain/OS)
<i>/usr/man/cat?/*</i>	Directories containing preformatted pages
<i>/usr/man/whatis</i>	Keyword database

SEE ALSO

apropos(1), *more(1)*, *whereis(1)*

NAME

msg – permit or deny messages

SYNOPSIS

msg [**n**] [**y**]

DESCRIPTION

msg with argument **n** forbids messages via **write** and **talk(1)** by revoking non-user write permission on the user's terminal. **msg** with argument **y** reinstates permission. All by itself, **msg** reports the current state without changing it.

FILES

/dev/tty*

/dev/display*

SEE ALSO

write(1), **talk(1)**

DIAGNOSTICS

Exit status is 0 if messages are receivable; 1 if not; 2 on error.

NAME

mk, **mkconv**, **membername** – maintain (make) related files

SYNOPSIS

mk [**-f** *mkfile*] ... [*option* ...] [*name* ...]

mkconv *makefile*

membername *aggregate* ...

DESCRIPTION

mk is most often used to keep object files current with the source they depend on.

mk reads *mkfile* and builds and executes dependency dags (directed acyclic graphs) for the target *names*. If no target is specified, the targets of the first non-metarule in the first *mkfile* are used. If no **-f** option is present, is tried. Other options are:

- a** Assume all targets to be out of date. Thus, everything gets made.
- d[egp]** Produce debugging output (**p** is for parsing, **g** for graph building, **e** for execution).
- e** Explain why each target is made.
- i** Force any missing intermediate targets to be made.
- k** Do as much work as possible in the face of errors.
- n** Print, but do not execute, the commands needed to update the targets.
- t** Touch (update the modified date of) non-virtual targets, without executing any recipes.
- u** Produce a table of clock seconds spent with *n* recipes running.
- wname1,name2,...**
Set the initial date stamp for each name to the current time. The names may also be separated by blanks or newlines. (Use with **-n** to find what else would need to change if the named files were modified.)

mkconv does a syntactic conversion from a *make(1)* *makefile* to a *mkfile* on standard output. Because of the semantic differences, the resulting *mkfile* may behave differently than the original *makefile*.

The shell script *membername* extracts member names (see ‘Aggregates’ below) from its arguments.

Definitions

A *mkfile* consists of *assignments* (described under ‘Environment’) and *rules*. A rule contains *targets* and a *tail*. A target is a literal string, or *label*, and is normally a file name. The tail contains zero or more *prerequisites* and an optional *recipe*, which is a shell script.

A *metarule* has a target of the form *A %B* where *A* and *B* are (possibly empty) strings. A metarule applies to any label that matches the target with **%** replaced by an arbitrary string, called the *stem*. In interpreting a metarule, the stem is substituted for all occurrences of **%** in the prerequisite names. The dependencies may refer to subexpressions in the normal way, using **\n**. The *dependency dag* for a target consists of *nodes* connected by directed *arcs*. A node consists of a label and a set of arcs leading to prerequisite nodes. The root node is labeled with an original target *name*.

Building the Dependency Dag

Read the *mkfiles* in command line order and distribute rule tails over targets to get single-target rules.

For a node *n*, for every rule *r* that matches *n*'s label generate an arc to a prerequisite node. The node *n* is then marked as done. The process is then repeated for each of the prerequisite nodes. The process stops if *n* is already done, or if *n* has no prerequisites, or if any rule would be used more than \$NREP times on the current path in the dag. A probable node is one where the label exists as a file or is a target of a non-metarule.

After the graph is built, it is checked for cycles, and subdags containing no probable nodes are deleted. Also, for any node with arcs generated by a non-metarule with a recipe, arcs generated by a metarule with a recipe are deleted. Disconnected subdags are deleted.

Execution

Labels have an associated date stamp. A label is *ready* if it has no prerequisites, or all its prerequisites are made. A ready label is *trivially uptodate* if it is not a target and has a nonzero date stamp, or it has a nonzero date stamp, and all its prerequisites are made and predate the ready label. A ready label is marked *made* (and given a date stamp) if it is trivially uptodate or by executing the recipe associated with the arcs leading from the node associated with the ready label. The **P** attribute can be used to generalise *mk*'s notion of determining if prerequisites predate a label. Rather than comparing date stamps, it executes a specified program and uses the exit status.

Date stamps are calculated differently for virtual labels, for labels that correspond to extant files, and for labels that don't. If a label is *virtual* (target of a rule with the **V** attribute), its date stamp is initially zero and when made set to the most recent date stamp of its prerequisites. Otherwise, if a label is nonexistent (does not exist as a file), its date stamp is set to the most recent date stamp of its prerequisites, or zero if it has no prerequisites. Otherwise, the label is the name of a file and the label's date stamp is always that file's modification date.

Nonexistent labels which have prerequisites and are prerequisite to other label(s) are treated specially unless the **-i** flag is used. Such a label *l* is given the date stamp of its most recent prerequisite. If this causes all the labels which have *l* as a prerequisite to be trivially uptodate, *l* is considered to be trivially uptodate. Otherwise, *l* is made in the normal fashion.

Two recipes are called identical if they arose by distribution from a single rule as described above. Identical recipes may be executed only when all their prerequisite nodes are ready, and then just one instance of the identical recipes is executed to make all their target nodes.

Files may be made in any order that respects the preceding restrictions.

A recipe is executed by *execing* the command

```
/bin/sh -ce recipe
```

The environment is augmented by the following variables:

\$alltarget	all the targets of this rule.
\$newprereq	the prerequisites that caused this rule to execute.
\$nproc	the process slot for this recipe. It satisfies $0 \leq \$nproc < \$NPROC$, where \$NPROC is the maximum number of recipes that may be executing simultaneously.
\$pid	the process id for the <i>mk</i> forking the recipe.
\$prereq	all the prerequisites for this rule.
\$stem	if this is a metarule, \$stem is the string that matched % . Otherwise, it is empty.
\$target	the targets for this rule that need to be remade.

Unless the rule has the **Q** attribute, the recipe is printed prior to execution with recognizable shell variables expanded. To see the commands print as they execute, include a *set -x* in your rule. Commands returning nonzero status cause *mk* to terminate.

Environment

Rules may make use of shell (or environment) variables. Any string which is a legal shell variable reference such as **\$OBJ** or **\${name}** is expanded. Variables can be set by assignments of the form

var = [*attr* =] *tokens*

where *tokens* and the optional attributes are defined below (under 'Syntax'). The environment is exported to recipe executions. Variable values are taken from (in increasing order of precedence) the below default values, the environment, the *mkfiles*, and any command line assignment. A variable assignment argument overrides the first (but not any subsequent) assignment to that variable.

```

AS=as                FFLAGS=                NPROC=1
CC=cc                LEX=lex                 NREP=1
CFLAGS=              LFLAGS=                YACC=yacc
FC=f77               LDFLAGS=               YFLAGS=
BUILTINS='
%.o:   %.c
        $CC $CFLAGS -c $stem.c
%.o:   %.s
        $AS -o $stem.o $stem.s
%.o:   %.f
        $FC $FFLAGS -c $stem.f
%.o:   %.y
        $YACC $YFLAGS $stem.y &&
        $CC $CFLAGS -c y.tab.c && mv y.tab.o $stem.o; rm y.tab.c
%.o:   %.l
        $LEX $LFLAGS -t $stem.l > $stem.c &&
        $CC $CFLAGS -c $stem.c && rm $stem.c'
ENVIRON=

```

The builtin rules are obtained from the variable **BUILTINS** after all input has been processed. The contents of the **ENVIRON** variable are split into parts separated by bytes with the **0200** bit set. These parts are then added to the environment without further processing. The variable **MKFLAGS** contains all the option arguments (arguments starting with a - or containing a =) and **MKARGS** contains all the targets in the call to *mk*.

Aggregates

Names of the form *a(b)* refer to member *b* of the aggregate *a*. Currently, the only aggregates supported are archives.

Syntax

Leading white space (blank or tab) is ignored. Input after an unquoted # (a comment) is ignored as are blank lines. Lines can be spread over several physical lines by placing a \ before newlines to be elided. Non-recipe lines are processed by substituting for *'cmd'* and then substituting for variable references. Finally, the filename meta-characters []*? are expanded. Quoting by ", "", and \ is supported. The semantics for substitution and quoting are given in *sh*(1).

Assignments and rule header lines are distinguished by the first unquoted occurrence of : (rule header) or = (assignment).

A rule definition consists of a header line followed by a recipe. The recipe consists of all lines following the header line that start with white space. The recipe may be empty. The first character on every line of the recipe is elided. The header line consists of at least one target followed by the rule separator : and a possibly empty list of prerequisites. The rule may have attributes specified in a : terminated list immediately following the : separator. The rule means if any prerequisite is more recent than any of the targets, the recipe is executed. This meaning is modified by the following attributes

- < The standard output of the recipe is read by *mk* as an additional mkfile. Assignments take effect immediately. Rule definitions are used when a new dependency dag is constructed.
- D If the recipe exits with an error status, the target is deleted.
- N If there is no recipe, the target has its time updated.
- P The characters after the P until the terminating : are taken as a program name. It will be invoked as *sh -c prog 'arg1' 'arg2'* and should return 0 exit status if and only if *arg1* is not out of date with respect to *arg2*. Date stamps are still propagated in the normal way.
- Q The recipe is not printed prior to execution.
- U The targets are considered to have been updated even if the recipe did not do so.
- V The targets of this rule are marked as virtual. They are distinct from files of the same name.

Similarly, assignments may have attributes terminated by a trailing =. The attributes are:

- U Do not export this variable to recipe executions. This may be necessary if you have many or large variables.

EXAMPLES

A simple mkfile to compile a program.

```
prog:  a.o b.o c.o
      $CC $CFLAGS -o $target $prereq
```

Override flag settings in the mkfile.

```
$ mk target CFLAGS='-O -s'
```

To get the prerequisites for an aggregate.

```
$ membername 'libc.a(read.o)' 'libc.a(write.o)' read.o write.o
```

Maintain a library.

```
libc.a(%o):N:%o
libc.a:  libc.a(abs.o) libc.a(access.o) libc.a(alarm.o) ...
        names='membername $newprereq'
        ar r libc.a $names && rm $names
```

Backquotes can be used to generate similar lists from a master list.

```
NAMES=alloc arc bquote builtins expand main match mk var word
OBJ='echo $NAMES|sed -e 's/^[^ ]*/&.o/g''
COBJ='echo $NAMES|sed -e 's/^[^ ]*/&.O/g''
```

A correct way to deal with *yacc*(1) grammars. The file *lex.c* includes the file *x.tab.h* rather than *y.tab.h* in order to reflect changes in content, not just modification time.

```
YFLAGS=-d
lex.o:  x.tab.h
x.tab.h: y.tab.h
        cmp -s x.tab.h y.tab.h || cp y.tab.h x.tab.h
y.tab.c y.tab.h:gram.y
        $YACC $YFLAGS gram.y
```

The above example could also use the **P** attribute for the *x.tab.h* rule:

```
x.tab.h:Pcmp -s:y.tab.h
        cp y.tab.h x.tab.h
```

SEE ALSO

make(1), *touch*(1), *sh*(1), *regex*(3)

BUGS

None of the conditional variable forms (such as **\${var:-word}**) are supported. Seemingly appropriate input like **CFLAGS=-DHZ=60** is correctly parsed as a (attribute) syntax error.

NAME

mkapr – make an Apollo product report

SYNOPSIS

mkapr [-v] [-c]

DESCRIPTION

The **mkapr** command creates a product report. This command replaces the **crucr** (create a user change request form) utility available in prior software releases.

Mkapr can be used to generate more than one product report without exiting and restarting the program. Please make a separate report for each problem or issue you have with an Apollo Computer product.

Output from **mkapr** may be in either (or both) of two forms:

1. Printed, human-readable copy; or
2. Encoded, transmittable form.

Printed product reports should be sent to:

APR Administrator/Customer Services
M/S CHM 02 GL
Apollo Computer Division of
Hewlett-Packard, Inc.
330 Billerica Road
Chelmsford, MA 01824

Encoded product reports may be sent to Apollo Customer Services via the UUCP network. The network address is:

apollo!apr_cs_admin

Recommended paths to Apollo are via **mit-eddie** or **decvax**. Some other sites that are short hops from Apollo are **uunet**, **kddlalab** and **unido**. These paths may change with time; check with your site administrator. If your site knows about one of the following Hewlett Packard sites, E-mail delivery to Apollo is rapid and reliable.

hplabs Palo Alto, CA
hp-sdd San Diego, CA
hp-pcd Corvallis, OR
hpbbn Boeblingen, West Germany
hplb Bristol, UK
hpycla Tokyo, Japan

Apollo is registered with the (ARPA) Internet. If your site has access, our Internet address is:

apr_cs_admin@apollo.hp.com

Customer Services will acknowledge all product reports received. Do not assume your product report has been received until you receive a reply. Security-conscious sites should not send confidential material. Voluminous submissions should be sent via magnetic media.

OPTIONS

Mkapr supports two options, **-v** and **-c**.

- v This will assert verbose mode; you will see information about **mkapr** actions such as locally saving copies of files and extended error and status messages. Normal mode operation is for **mkapr** to invoke the system services silently. **mkapr** always allows system services to report errors directly to the standard error device.
- c This will cause **mkapr** to start up with the command driven interface rather than with the DIALOG graphic interface.

SERVICES SUPPORTED

In addition to creating Apollo Problem Reports online, **mkapr** will make available viewing, editing, printing and mailing services if they exist (and **mkapr** can find them). The mailing services known to **mkapr** are:

Unix environment - sendmail
 Aegis environment - DPSS/mail (ONLY if no sendmail)

You tell **mkapr** how to make use of **sendmail** by specifying a value for the MAIL_OPTIONS entry in the **.aprint** initialization file. **Mkapr** uses the options '-om -oi' by default. See the manual page for sendmail(8) for an explanation of the available options.

If sendmail is not available, and DPSS/mail is available, **mkapr** will use DPSS/mail. In this case, the MAIL_OPTIONS entry in the **.aprint** file is ignored.

The print services known to **mkapr** are:

All environments - prf
 BSD environment - lpr
 SysV environment - lp

You tell **mkapr** how to print by specifying a value for the PRINT_COMMAND entry in the **.aprint** initialization file. The default print command used by **mkapr** is

prf -left 0.5 -wrap

If a desired service is not available to **mkapr**, a product report (print or send) file will be saved in the current working directory for printing or sending at a later time.

DIALOG INTERFACE

Mkapr will make use of the DIALOG graphic interface environment of the Apollo Domain system whenever possible. This interface is intended to be easy to learn and use. It is active when the cursor (arrow) is in the **mkapr** display window. Pressing the F2 function key or the middle mouse button will display a brief help message that will go away when the key or button is released.

It will be useful to you to review the command action descriptions in the COMMAND DRIVEN INTERFACE section below. There is a direct correspondence between action buttons in the graphic interface and commands in the line-oriented interface.

If you select a field in the DIALOG graphic interface for editing (by point-and-click-with-M1/F1-or-M3/F3), a small filled triangle appears in the field as near to the cursor arrow as possible. This marks the insertion point for text.

If you type in the field, light square brackets appear at the far left and far right of the field (they can be hard to see). While those brackets are present, **mkapr** has not yet seen the text you have typed. Any Print, Send, or View action will use the field values known to **mkapr** (most likely these are empty string values).

If you press the <RETURN> key, all text from the left margin of the text field to the text insertion point (triangle) is passed to **mkapr**'s internal service routines for processing. The square brackets disappear to indicate that the editing of the field has completed.

There is no limit to the number of pending field edits. By starting a field edit, then pressing <TAB> to proceed to the next field (you think you have finished the edit), and starting to type in the new field, you now have two field edits underway. The edits must be completed with the <RETURN> key.

The <TAB> key is intended for quickly moving from one field to another without changing the field value. The <RETURN> key will move from one field to another, but will have drastic effects on the field values.

The <BACKSPACE> key will delete the character to the left of the text insertion point.

The <CHAR DEL> key will delete the character to the right of the text insertion point.

The '|<-' and '->|' keys will move the text insertion point to the beginning and end, respectively, of the text edit field.

The editable text fields in the graphic interface have a pop-up help box which summarizes the preceding editing information.

COMMAND DRIVEN INTERFACE

If the display environment you are using does not support the graphic interface, you will see the following prompt:

```
mkapr>
```

Entering the command 'help' will display the available commands. Here is the list of commands for reference:

Command	Description
help [mkapr]	List Commands. To display the help file, use the mkapr option.
change	Change APR Information Fields.
edit	Edit the detailed Problem Description Section.
view	View the current APR Information.
print	Print the current APR Information using PRINT_COMMAND.
send	Send the current APR Information to MAIL_PATH with MAIL_OPTIONS.
save	Save the current APR Print and Send files in the SAVE_PATH.
exit	Save current customer information changes (if any) and exit.

another Create another Product Report without exiting **mkapr**.

You need only enter as much of any command as is necessary to uniquely identify it. For example, you need only type **ch** for the change command. Detailed descriptions of the commands follow.

change This allows you to provide the information needed to complete a product report. Some information is extracted automatically from the system **mkapr** is running on. Some information comes from an initialization file, **.aprint**. You may change most of these default values. Other information you must provide yourself.

Within the change command, the prompt becomes

mkapr..change>

All current input values are displayed by field. Fields whose values can't be changed are flagged with an asterisk (*). You will be asked to enter the field number to change, then asked to enter the new value (pressing <RETURN> will leave the field value unchanged). The cycle is then repeated. If you reply 'h' or 'help' at this point, you will get the following help message for the change command:

Change Command Description

help [mkapr] List commands. To display the help file, use the **mkapr** option.

display fields Display all fields and their respective values.

change field n Request to change the value of field # *n*. Pressing the RETURN key at the prompt

enter new value ==>

will leave the value unchanged.

exit Exit the change command.

edit An editor will be invoked using available system services. You should enter a detailed problem description, save it, and exit the editor using the appropriate editor commands. You will then be returned to the **mkapr>** prompt.

view The current **mkapr** information will be displayed using available system services.

print The current **mkapr** information will be printed to the default printer using the PRINT_COMMAND value specified in the **.aprint** initialization file.

In a BSD4.3 environment, the shell environment variable **PRINTER** can be used to indicate a printer to use in place of the default assumed by the `/usr/ucb/lpr` command. In a SYS5.3 environment, the shell environment variable **LPDEST** is used by the `/usr/bin/lp` command. In an SR10 Aegis environment, the file `((HOME))/user_data/startup.prf` may be used to provide needed information to the `/usr/apollo/bin/prf` command. In a pre-SR10 Aegis environment, the file `((HOME))/user_data/prf.db` serves the same purpose.

send The current **mkapr** information will be sent to the destination specified by the **MAIL_PATH** entry in the **.aprinit** file using `/usr/lib/sendmail` and the **MAIL_OPTIONS** entry in the **.aprinit** file. If `/usr/lib/sendmail` is unavailable and **DPSS/mail** is available, then **DPSS/mail** will be used as a last resort. The **MAIL_OPTIONS** entry is ignored in this case.

save **mkapr** will create the output files used by the **PRINT** and **SEND** actions (print files are human-readable, send files are for e-mail transmission). These files are initially created in the `/tmp` temporary directory; they are then moved to the directory specified by the value of the **SAVE_PATH** entry in the **.aprinit** initialization file. The file names are:

```
apr.<APR_ID>.p      the print file
apr.<APR_ID>.s      the send  file
```

The default value for **SAVE_PATH** is `'.'` (the current working directory).

exit If any changes to customer information occurred during this session, save all customer information to the non-system **.aprinit** file, delete all temporary files created by **mkapr** during this session (temporary files belonging to other **mkapr** sessions are ignored), then exit **mkapr**.

another Create another product report without leaving **mkapr**. Be sure you have done something appropriate with the current product report before issuing this command (Send, Print, Save or View). This command resets all product report field values to their initial values, except for the customer contact, name, address, and telephone number fields. A new **APR_ID** value is generated.

INITIAL FIELD VALUES

The fields of an Apollo Problem Report that are collectively known as customer information Fields are initialized from a file read when **mkapr** starts up. These fields contain such information as the name of the customer contact, the name (company name) of the customer, and the customer's address and telephone number. The initialization file has the name **.aprinit** and the **mkapr** program will search for it. The search order for the initialization file is:

1. Look in the current working directory
2. Look in the home directory as given by the shell variable HOME
3. Look in the system directory /etc/apr

It is not an error for no initialization file to exist; **mkapr** will leave the customer information fields blank. The fields can be edited and the initialization file will be updated when **mkapr** exits.

The file **/etc/apr.aprinit** is a special case; **mkapr** will not write to this file. The system administrator (or other privileged account) must create the directory **/etc/apr** with appropriate access permissions, then run **mkapr** to create a local copy of the file **.aprinit** and copy or move the file to the directory.

The initialization file is an ASCII text file that may be created and modified using any of the text editors available to you. The body of the **.aprinit** file created by **mkapr** is reproduced here:

```
# .aprinit aruser 15 August 1989
# Comment lines begin with '#'
# Non-comment lines have the following form:
# FIELD_NAME : FIELD_VALUE : IGNORED
# The field name must not be changed.
# The ':' character delimits fields.
# The field value may be changed; it must not contain ':'.
# unless the field value is quoted by either ' ' or " " pairs.
# Anything after the second ':' is thrown away.
#
customer_contact : A. Random User
customer_name : Apollo Computer Division of Hewlett-Packard, Inc.
customer_addr1 : CHM 02 GL
customer_addr2 : 330 Billerica Road
customer_addr3 : Chelmsford, MA 01824
customer_addr4 : USA
customer_phone : 1-508-256-6600
save_path : .
mail_path : apr_cs_admin@apollo.hp.com
mail_options : -oi -om
print_command : prf -left 0.5 -wrap
#
# End of .aprinit
```

NOTES

Since **mkapr** assumes that the site mail facility (probably **sendmail**) knows how to get from your site to Apollo, you must edit the **MAIL_PATH** field value in **.aprinit** to give **mkapr** the correct path. Be sure that your mail facility is set up correctly. See your site administrator for help.

Run `/usr/ucb/newaliases` at least once before attempting to use **mkapr**'s send function. **Mkapr** will warn you if the **sendmail** database files are either inconsistent or don't exist when it starts up.

Offsite mailing may not be allowed by your site. If so, you must make other arrangements to get mail to Apollo. See your site administrator for help.

FILES

<code>/usr/apollo/bin/mkapr</code>	The SR10 executable COFF object
<code><AA>/install/sr9.7_compatibility/sr9.7_executables/com/mkapr</code>	The SR9.7 OBJ version
<code>/usr/man/cat1/mkapr.1</code>	This manual page (UNIX BSD)
<code>/usr/catman/u_man/man1/mkapr.1</code>	This manual page (UNIX SysV)
<code>/sys/help/mkapr.hlp</code>	This help file (Aegis)
<code>/sys/dm/fonts/vt100l.*</code>	Display fonts
<code>/sys/dialog/dp_*</code>	Display fonts
Initialization file (search order):	
<code>.aprint</code>	(1st) (updated)
<code>\$HOME/.aprint</code>	(2nd) (updated)
<code>/etc/apr/.aprint</code>	(last) (read only)
<code>/tmp/apr.*</code>	Temporary files:
<code>apr.*.v.*</code>	Product report view file
<code>apr.*.p.*</code>	Product report print file
<code>apr.*.s.*</code>	Product report send file
<code>apr.*.c.*</code>	Product report send command file
<code>apr.*.e.*</code>	Problem description edit file

NAME

mkdir – make a directory

SYNOPSIS

mkdir *dirname*...

DESCRIPTION

mkdir creates specified directories in mode 777. It makes standard entries, '.' for the directory itself and '..' for its parent, automatically.

mkdir requires write permission in the parent directory.

SEE ALSO

rmdir(1)

NAME

mkstr – create an error message file by massaging C source

SYNOPSIS

mkstr [-] *messagefile prefix file ...*

DESCRIPTION

mkstr is used to create files of error messages. It can make programs with many error diagnostics much smaller, and can reduce system overhead in running the program because error messages do not have to be constantly swapped in and out.

mkstr processes each specified *file*, placing a massaged version of the file in a file whose name consists of the specified *prefix* added to the original name.

To process the error messages in the source to the message file, **mkstr** keys on the string 'error("' in the input stream. **mkstr** puts each C string, starting at the quotation mark, in the message file, followed by a null character and a newline character. The null character terminates the message so you can use it easily when you retrieve it. The newline character allows you to sensibly **cat** the error message file to see its contents. The massaged copy of the input file then contains an **lseek** pointer into the file, that you can use to retrieve the message, for example,

```
char    efilename[] = '/usr/lib/pi_strings';
int     efil = -1;

error(a1, a2, a3, a4)
{
    char buf[256];

    if (efil < 0) {
        efil = open(efilename, 0);
        if (efil < 0) {
oops:
            perror(efilename);
            exit(1);
        }
    }
    if (lseek(efil, (long) a1, 0) || read(efil, buf, 256) <= 0)
        goto oops;
    printf(buf, a2, a3, a4);
}
```

If you specify the dash (-), **mkstr** puts error messages at the end of the specified message file for recompiling part of a large **mkstr** program.

EXAMPLE

Specify the following command if you want to put all the error messages from the C source files in the current directory into the file *pistrings*, and put processed copies of the source for these files in files whose names are prefixed with *xx*:

```
mkstr pistrings xx *.c
```

SEE ALSO

lseek(2), xstr(1)

NAME

more, **page** – file perusal filter for CRT viewing

SYNOPSIS

```
more [ -cdfisu ] [ -n ] [ +linenumber ] [ +/pattern ] [ name ... ]
page [ more-options ]
```

DESCRIPTION

more is a filter that lets you examine continuous text, one full screen at a time, on a soft-copy terminal. It normally pauses after each full screen and displays "--More--" at the bottom of the screen. If you respond by entering a newline, **more** displays one more line. If you press the spacebar, it displays the next screen of text.

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 pre-set any flags desired. For example, if you prefer to view files via the **-c** mode of operation, you can use the **sh(1)** command

```
setenv MORE -c
```

or the **sh(1)** command sequence

```
MORE='-c' ; export MORE
```

to force all invocations of **more**, including those by programs such as **man(1)**, and **msgs(1)**, to use this mode. Normally, you place the command sequence that sets up the **MORE** environment variable in the **.cshrc** or **.profile** file.

If **more** is reading from a file, rather than from a pipe, it reports, on the same line as the "--More--" prompt, the percentage of the total text (in characters, not lines) displayed on the screen so far.

If the standard output is not a tty, **more** behaves as **cat(1)** does, except that it displays a header before each file.

If you run **more** from a Display Manager window, a vt100 emulator is started first.

page clears the screen before displaying each full screen (only if a full screen is to be displayed), and displays $k - 1$ rather than $k - 2$ lines on each full screen, where k is the number of lines the terminal can display.

OPTIONS

- n** Set the size (in lines) of the display window to n .
- c** Draw each page by beginning at the top of the screen and erasing each line just before drawing on it. This avoids scrolling the screen, making text easier to read while **more** is writing. **more** ignores this option if the terminal cannot clear to the end of a line.

- d Prompt with the message "Press space to continue, 'q' to quit." at the end of each full screen. Respond to subsequent illegal user input by printing "Press 'h' for instructions." instead of ringing the bell. This is helpful if you use **more** as a filter in some setting involving unsophisticated users.
- f Count logical lines rather than screen lines. Do not fold long lines. This option helps when you pipe **nroff**(1) output through **ul**(1), since the latter may generate escape sequences. These escape sequences contain characters that ordinarily occupy screen positions, but are not displayed when sent to the terminal as part of an escape sequence. Thus, **more** may interpret lines to be longer than they actually are, and fold lines erroneously.
- l Do not treat form feeds specially. Without this option, **more** pauses after any line containing a form feed, as if the end of a full screen had been reached. Also, if a file begins with a form feed, **more** clears the screen before displaying the file.
- s Squeeze multiple blank lines from the output, producing only one blank line. This option maximizes the useful information present on the screen; it is especially helpful when you are viewing **nroff**(1) output.
- u Suppress the normal method of handling underlining. Thus, do not output appropriate escape sequences to enable underlining or standout mode for underlined information in the source file.
- +*linenumber*
Begin displaying text at *linenumber*.
- +/*pattern*
Begin displaying text two lines before the line containing the regular expression *pattern*.

COMMANDS

Other sequences that you can type when **more** pauses are described below (*i* is an optional integer argument, defaulting to 1). The commands take effect immediately, that is, you do not have to enter a carriage return. Up to the time you specify the command character itself, you can use the line-kill character to cancel the numerical argument being formed. You can also use the erase character to redisplay the "--More--(xx%)" message.

You can issue an interrupt any time output is being sent to the terminal. **more** stops sending output and displays the usual "--More--" prompt. You can then enter one of the commands in the normal manner. Unfortunately, you lose some output when you do this, because any characters waiting in the terminal's output queue are flushed when you issue the quit signal.

more sets the terminal to noecho mode, so that the output can be continuous. Thus, what you type does not appear on the terminal, except when you use slash (/) or exclamation point (!) commands.

- i* <space> Display *i* more lines (or another full screen, if you do not specify an argument).

CTRL/D	Display 11 more lines (a "scroll"). If you also specify <i>i</i> (iCTRL/D), set the scroll size to <i>i</i> .
d	This command is the same as D (CTRL/D).
i z	This is the same as typing a space, except that <i>i</i> , if present, becomes the new window size.
i s	Skip <i>i</i> lines ahead and print the next full screen of lines after that.
i f	Skip <i>i</i> full screens ahead and print the next full screen of lines after that.
i b	Skip <i>i</i> full screens back and print the next full screen of lines.
i CTRL/B	This is the same as <i>i b</i> .
q or Q	Exit from the program (more or page).
=	Display the current line number.
v	Start up the editor vi (1) at the current line.
h	Help; give a description of all the more commands.
i/expr	Search for the <i>i</i> th occurrence of the regular expression <i>expr</i> . If there are fewer than <i>i</i> occurrences of <i>expr</i> , and the input is a file rather than a pipe, the position in the file remains unchanged. Otherwise, more displays a full screen, starting two lines before the location of the expression. You can use erase and kill characters to edit the regular expression. Erasing back past the first column cancels the search command.
i n	Search for the <i>i</i> th occurrence of the last regular expression entered.
` (single quote)	Go to the point where the last search began. If no search was performed in the current file, go to the beginning of the file.
!command	Invoke a shell with <i>command</i> . Replace the percent (%) and exclamation point (!) characters in <i>command</i> with the current filename and the previous shell command, respectively. If no current filename exists, the percent (%) is not expanded. The \% and \! sequences are replaced by % and !, respectively.
i:n	Skip to the <i>i</i> th next file given in the command line (or to last file if <i>n</i> is inappropriate).
i:p	Skip to the <i>i</i> th previous file given in the command line. If you issue this command while printing a file, more returns to the beginning of the file. If <i>i</i> is inappropriate, more skips back to the first file. If more is not reading from a file, it beeps the terminal, but takes no further action.
:f	Display the current filename and line number.
:q or :Q	This is the same as q or Q.

. (dot) Repeat the previous command.

EXAMPLE

To use **more** to help preview **nroff(1)** output, type the following:

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

FILES

/etc/termcap	Terminal database
/usr/lib/more.help	Help file

BUGS

Skipping backwards is too slow on large files.

SEE ALSO

csh(1), **crpad(1)**, **msgs(1)**, **man(1)**, **sh(1)**, **environ(7)**

NAME

msgs - system messages and junk mail program

SYNOPSIS

```
msgs [ -fhlpq ] [ number ] [ -number ]  
msgs -s  
msgs -c [ -days ]
```

DESCRIPTION

msgs is used to read system messages. These messages are sent by mailing to the login 'msgs' and should be short pieces of information, typically read once by most system users.

Normally you invoke **msgs** at login by placing the command in the file **.login** (**.profile** if you use **/bin/sh**).

msgs keeps message information in the file **.msgsrc** in your home directory. It keeps a set of files whose names are the (sequential) numbers of the messages they represent in the directory **/usr/msgs**. The file **/usr/msgs/bounds** shows the low and high number of the messages in the directory so that **msgs** can quickly determine if there are no messages for you. If the contents of **bounds** are incorrect, remove the file; **msgs** will make a new **bounds** file the next time it runs.

COMMANDS

msgs prompts you with the source and subject of each new message. If there is no subject line, it displays the first few non-blank lines of the message. If there is more to the message, **msgs** tells you how long it is and asks whether you wish to see the rest of the message. The possible responses are

y (or **<RETURN>**) Display the rest of the message.

n Skip this message and go on to the next.

- Redisplay the last message.

q Exit **msgs**; resume examination of the message queue at this point when the program is run again.

s (or **s-**) Append the current message to the file **Messages** in the current directory. **s-** saves the previously displayed message. You can follow either **s** or **s-** with a space and a filename to receive the message.

m (or **m-**) Place a copy of the specified message in a temporary mailbox and invoke **mail(1)** on that mailbox. Both **m** and **s** accept a numeric argument in place of the dash (**-**).

You can also go to any specific message by typing its number when **msgs** requests input.

OPTIONS

- f Suppress printing of "No new messages."
- h Display only the first part of messages.
- l Report only locally originated messages.
- p Pipe long messages through `more(1)`.
- q Check for new messages only. If found, display "There are new messages." This option is often used in log-in scripts.
- number* Start at the specified message rather than at the next message indicated by your `.msgsrc` file.
- number* Start *number* messages back from the one indicated by your `.msgsrc` file. This option is useful for reviewing recent messages.
- s Sets up the posting of messages. You should include the line
`msgs: "| /usr/ucb/msgs -s"`
in `/usr/lib/aliases` to enable posting of messages.
- c Performs cleanup on `/usr/msgs`. You should place an entry with the `-cn` option in `/usr/lib/crontab` to run every night and remove all messages over *n* days old. The default is 21 days.

FILES

<code>/usr/msgs/*</code>	Database
<code>~/msgsrc</code>	Number of next message to be presented

SEE ALSO

`aliases(5)`, `crontab(5)`, `mail(1)`, `more(1)`

NAME

mt – magnetic tape manipulating program

SYNOPSIS

mt [**-f** *tapename*] [**-scsl**] *command* [*count*]

DESCRIPTION

mt gives commands to a magnetic tape drive. If you do not specify a tape name, **mt** uses the environment variable **TAPE**. If **TAPE** does not exist, the default device is **/dev/rmt12**. Note that *tapename* must reference a raw (not block) tape device. By default **mt** performs the requested operation once. Specify *count* if you want to perform operations more than once.

You must use the **-scsl** option when *tapename* refers to a SCSI device. **rewind** is the only command available with this option.

COMMANDS

You need to specify only as many characters as needed to uniquely identify a command.

eof, weof

Write *count* end-of-file marks at the current position on the tape.

fsf Forward space *count* files.

fsr Forward space *count* records. This command cannot be used with cartridge tapes.

bsf Backspace *count* files. This command cannot be used with cartridge tapes.

bsr Backspace *count* records. This command cannot be used with cartridge tapes.

erase Erase from current position to end-of-tape.

reten Retension cartridge tapes.

rewind Rewind the tape. Ignore *count*.

offline, rewoffl

Rewind the tape and place the tape unit offline. Ignore *count*.

status Print status information about the tape unit.

DIAGNOSTICS

returns a 0 exit status when the operations are successful; 1 if a command is unrecognized; and 2 if an operation fails.

FILES

/dev/rct* Raw cartridge tape interface

/dev/rmt* Raw magnetic tape interface

SEE ALSO

ctio(4), **mtio(4)**, **dd(1)**, **ioctl(2)**, **environ(7)**

NAME

mv – move or rename files

SYNOPSIS

mv [**-i**] [**-f**] [**-**] *file1 file2*

mv [**-i**] [**-f**] [**-**] *file ... directory*

DESCRIPTION

mv moves (changes the name of) *file1* to *file2*.

If *file2* already exists, **mv** removes it before moving *file1*. If *file2* has a mode that forbids writing, **mv** prints the mode and reads the standard input to obtain a line. If the line begins with **y**, **mv** executes the move; if not, **mv** exits. See **chmod(2)** for more information about modes.

In the second form, **mv** moves one or more *files* (plain files or directories) to the *directory*, with their original filenames.

mv refuses to move a file onto itself.

OPTIONS

- i** Operate in interactive mode. Whenever a move would supersede an existing file, **mv** prompts you with the filename followed by a question mark. If you answer with a line starting with **y**, **mv** continues. Any other reply prevents the move from occurring.
- f** Override any mode restrictions or the **-i** option (force).
- Interpret all the following arguments to **mv** as filenames. This allows you to specify filenames starting with minus.

BUGS

If *file1* and *file2* are 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 the file loses any linking relationship with other files.

SEE ALSO

cp(1), **ln(1)**

NAME

nawk – pattern scanning and processing language

SYNOPSIS

nawk [**-Fre**] [*parameter*] ['*prog*'] [**-f progfile**] [*file* . . .]

DESCRIPTION

nawk is a new version of **awk** that provides capabilities unavailable in previous versions. This version will become the default version of **awk** in the next major UNIX system release.

nawk scans each input *file* for lines that may be a set of patterns specified in *prog*. With each pattern in *prog* 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 quotes (') to protect it from the shell.

The **-Fre** option defines the input file separator to be the regular expression *re*.

Parameters, in the form *x=... y=...* etc., may be passed to **nawk**.

Input 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**; see below). The fields are denoted **\$1**, **\$2**, ...; **\$0** refers to the entire line.

Built-in variables include:

ARGC	Command line argument count.
ARAGV	Command line argument array.
FILENAME	Name of the current input file.
FNR	Ordinal number of the current record in the current file.
FS	Input field separator regular expression (default blank).
NF	Number of fields in the current record.
NR	Ordinal number of the current record.
OFMT	Output format for numbers (default % .6g).
OFS	Output field separator (default blank).
ORS	Output record separator (default new-line).
RS	Input record separator (default new-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
do statement while ( conditional )
for ( expression ; conditional ; expression ) statement
for ( var in array ) statement
delete array[subscript]
break
continue
{ [ statement ] ... }
variable = expression
print [ expression-list ] [ >expression ]
printf format [ , expression-list ] [ >expression ]
next
exit [ expr ]
return [ expr ]

```

Statements are terminated by semicolons, new-lines, 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 *>expr* 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(3S)* in the *BSD Programmer's Reference*].

nawk has a variety of built-in functions: arithmetic, string, input/output, and general.

The arithmetic functions are: **atan2**, **cos**, **exp**, **int**, **log**, **rand**, **sin**, **sqrt**, and **srand**. **int** truncates its argument to an integer. **rand** returns a random number between 0 and 1. **srand** (*expr*) sets the seed value for **rand** to *expr* or uses the time of day if *expr* is omitted.

The string functions are:

gsub(*for*, *repl*, *in*)

Behaves like **sub** (see below), except that it replaces successive occurrences of the regular expression (like the *ed* global substitute command).

index(*s*, *t*)

Returns the position in string *s* where string *t* first occurs, or 0 if it does not occur at all.

length(*s*)

Returns the length of its argument taken as a string, or of the whole line if there is no argument.

match(*s*, *re*)

Returns the position in string *s* where the regular expression *re* occurs, or 0 if it does not occur at all. **RSTART** is set to the starting position (which is the same as the returned value), and **RLENGTH** is set to the length of the matched string.

split(*s*, *a*, *fs*)

Splits the string *s* into array elements *a*[1], *a*[2], *a*[*n*], and returns *n*. The separation is done with the regular expression *fs* or with the field separator **FS** if *fs* is not given.

sprintf(*fmt*, *expr*, *expr*, ...)

Formats the expressions according to the **printf(3S)** format given by *fmt* and returns the resulting string.

sub(*r*, *s*)

Substitutes the string *s* for first *r* in current record; returns number of substitutions.

substr(*s*, *m*, *n*)

Returns the *n*-character substring of *s* that begins at position *m*.

The input/output and general functions are:

close(*filename*)

Closes the file or pipe named *filename*.

cmd/getline

Pipes the output of **cmd** into **getline**; each successive call to **getline** returns the next line of output from **cmd**.

getline Sets **\$0** to the next input record from the current input file.

getline < *file*

Sets **\$0** to the next record from *file*.

getline *var*

Sets variable *var* instead.

getline *var* < *file*

Sets *var* from the next record of *file*.

system(*cmd*)

Executes *cmd* and returns its exit status.

All forms of **getline** return 1 for successful input, 0 for end-of-file, and -1 for an error.

nawk also provides user-defined functions. Such functions can be defined (in the pattern position of a pattern-action statement) as:

```
function name(args,...) { stmts }
func name (args,...) { stmts }
```

Function arguments are passed by value if scalar and by reference if array name. Argument names are local to the function; all other variable names are global.

Function calls can be nested and functions can be recursive. The return statement can be used to return a value.

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** (see **grep(1)**). 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 output field separator (default blank); ORS, output record separator (default new-line); 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 }
```

Same, but with input fields separated by comma and/or blanks and tabs:

```
BEGIN {FS = ",[ ]*[" ]+" }
{ print $2, $1 }
```

Add 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 }
```

Print file, filling in page numbers starting at 5:

```
/Page/ { $2 = n++; }
{ print }
```

command line: **nawk -f program n=5 input**

BUGS

Input white space is not preserved on output if fields are involved.

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 the null string (" ") to it.

SEE ALSO

grep(1), sed(1).

lex(1), printf(3S) in the *BSD Programmer's Reference*.

nawk in the *BSD UNIX User's Manual*

NAME

netstat – show network status

SYNOPSIS

```
netstat [ -Aang ] [ -f address_family ]
netstat [ -himnrstT ] [ -f address_family ]
netstat [ -n ] [ -I interface ] interval
```

DESCRIPTION

The **netstat** command symbolically displays the contents of various network-related data structures. You can specify one of a number of output formats. The first form of the command displays a list of active sockets for each protocol. The second form presents the contents of one of the other network data structures according to the option selected. The third form, with an *interval* specified, continuously displays the information regarding packet traffic on the configured network interfaces.

The default display, for active sockets, shows the local and remote addresses, send and receive queue sizes (in bytes), protocol, and the internal state of the protocol. Address formats are of the form *host.port* or *network.port* if a socket's address specifies a network but no specific host address. It displays the host and network addresses when known, symbolically, according to the databases */etc/hosts* and */etc/networks*, respectively, or from the databases maintained by */etc/named*. If a symbolic name for an address is unknown, or if you specify the **-n** option, **netstat** displays the address numerically, according to the address family. For more information regarding the Internet "dot format," refer to **inet(3N)**. **netstat** displays unspecified, or "wildcard", addresses and ports an asterisk (*).

The interface display provides a table of cumulative statistics regarding packets transferred, errors, and collisions. It also shows the network addresses of the interface and the maximum transmission unit (**mtu**).

The routing table display indicates the available routes and their status. Each route consists of a destination host or network and a gateway to use in forwarding packets.

The flags field shows the following:

- The state of the route (U if "up")
- Whether the route is to a gateway (G)
- Whether the route was created dynamically by a redirect (D)
- Whether the route has priority (P)
- Whether the route is a static (S) route added with **route**
- Whether the route has been marked for deletion (X)

Direct routes are created for each interface attached to the local host; the **gateway** field for such entries shows the address of the outgoing interface. The **refcnt** field gives the current number of active uses of the route. Connection oriented protocols normally hold on to a single route for the duration of a connection while connectionless protocols obtain a route while sending to the same destination. The **use** field provides a count of the number of packets sent using that route. The interface entry indicates the network interface utilized for the route.

When you invoke **netstat** with an *interval* argument, it displays a running count of statistics related to network interfaces. This display consists of a column for the primary interface (the first interface found during auto-configuration) and a column summarizing information for all interfaces. Use the **-I** option to replace the primary interface with another interface. The first line of each screen of information contains a summary since the system was last rebooted. Subsequent output lines show values accumulated over the preceding interval.

OPTIONS

- A** With the default display, show the address of any protocol-control blocks (PCB) associated with sockets; used for debugging. If a 0 appears in the first (PCB) column, it means that a socket is in the process of being shut down.
- a** With the default display, show the state of all sockets; normally sockets used by server processes are not shown.
- g** With the default display, show the first gateway used.
- h** In the Domain/OS implementation, shows the state of the ARP host table. Status flags show whether the entry is for a gateway (G), is in use (U), or is temporary (T). When you use the **-h** option on a host that has the IEEE 802.5 network interface enabled, the default output displays an additional field called LanInfo. This field reports the broadcast indicator, length, direction, and frame size of the 802.5 Media Access Control (MAC) frame.
- i** Show the state of interfaces that were auto-configured (**netstat** does not show interfaces statically configured into a system, but not located at boot time).
- I *interface*** Show information only about this interface; used with an *interval* as described below.
- m** Show statistics recorded by the memory-management routines (the network manages a private pool of memory buffers).
- n** Show network addresses as numbers (normally **netstat** interprets addresses and attempts to display them symbolically). You can use this option with any of the display formats.
- s** Show per-protocol and routing statistics.
- r** Show the routing tables.
- t** When used with the **-i** option, show timer column.
- T** Show all possible status information.
- f *address_family*** Limit statistics or address-control-block reports to those of the specified *address_family*. The following address families are recognized: **inet**, for AF_INET; **ns**, for AF_NS; and **unix**, for AF_UNIX.

BUGS

The notion of errors is ill-defined.

SEE ALSO

hosts(5), networks(5), protocols(5), services(5), trpt(8C)

NAME

newaliases – rebuild the database for the mail aliases file

SYNOPSIS

newaliases

DESCRIPTION

newaliases rebuilds the random access database for the mail aliases file `/usr/lib/aliases`. You must run it each time `/usr/lib/aliases` is changed in order for the change to take effect.

newaliases is a link to `/usr/lib/sendmail`.

FILES

`/usr/lib/aliases` Aliases list

SEE ALSO

`aliases(5)`, `sendmail(8)`

NAME

nice, **nohup** – run a command at a changed priority (**sh** only)

SYNOPSIS

nice [*-number*] *command* [*arguments*]

nohup *command* [*arguments*]

DESCRIPTION

nice executes *command* with low scheduling priority. If you specify *number*, **nice** increments the priority (higher numbers mean lower priorities) by that amount up to a limit of 20. The default *number* is 10.

A user can run commands with priority higher than normal by using a negative priority, for example, “--10”.

nohup executes *command*, making it immune to hang-up and terminate signals from the controlling terminal. The priority is incremented by five. You should invoke **nohup** from the shell with a trailing ampersand (&) in order to prevent it from responding to interrupts by or stealing the input from the next person who logs in on the same terminal.

FILES

nohup.out Standard output and standard error file under **nohup**

SEE ALSO

csh(1), **setpriority**(2), **renice**(8)

DIAGNOSTICS

nice returns the exit status of the subject command.

BUGS

nice and **nohup** are particular to **sh**(1). If you use **csh**(1), commands executed with “&” are automatically immune to hang-up signals while in the background. A built-in command, **nohup**, provides immunity from terminate, but does not redirect output to **nohup.out**.

nice is built into **csh**(1) with a slightly different syntax than described here. The form “**nice +10**” runs a command at low priority, and the super-user can use “**nice -10**” to give a process more of the processor.

Because Domain/OS priority scheduling uses a different scale (1 to 16 instead of 0 to 20), there may be rounding problems, and processes may be “niced” to a slightly different value than expected.

NAME

nm - print name list

SYNOPSIS

nm [**-agnopru**] [*file* ...]

DESCRIPTION

nm prints the name list (symbol table) of each object *file* in the argument list. If an argument is an archive, **nm** produces a listing for each object file in the archive. If you do not specify a *file*, **nm** lists the symbols in the file **a.out**.

Each symbol name is preceded by its value (blanks if undefined) and one of the letters U (undefined), A (absolute), T (text-segment symbol), D (data-segment symbol), B (bss segment symbol), C (common symbol), S (user defined segment symbol), *f* filename, - for debugger symbol table entries (see **-a** below), or G (global library symbol) if the Domain/OS BSD option, **-Ag**, is supplied. If the symbol is local (non-external) the type letter is lowercase. **nm** sorts its output alphabetically.

OPTIONS

- a** Print symbol table entries inserted for use by debuggers.
- g** Print only global (external) symbols.
- n** Sort numerically rather than alphabetically.
- o** Prefix file or archive element name to each output line.
- p** Don't sort; print in symbol-table order.
- r** Sort in reverse order.
- u** Print only undefined symbols.
- Ag** Check KGT (Known Global Table) to see if undefined globals are defined in global libraries. If specified with the **-u** option, **nm** will not print those undefined symbols that are defined in global libraries.

SEE ALSO

ar(1), **ar(5)**, **a.out(5)**

NAME

nroff – text formatting

SYNOPSIS

nroff [*options*] ... [*file* ...]

DESCRIPTION

nroff formats text in the named *files* for printing on typewriter-like devices.

If you do not specify a *file*, or if you specify a single dash (-), **nroff** reads the standard input.

OPTIONS

You can specify the options in any order provided you put them before the files.

- olist** Print only pages whose page numbers appear in the comma-separated *list* of numbers and ranges. A range *N-M* means pages *N* through *M* inclusive; an initial *-N* means from the beginning to page *N*; and a final *-* means from *N* to the end.
- nN** Number the first generated page *N*.
- sN** Stop every *N* pages. **nroff** halts before every *N*th page (the default is 1) to allow paper loading or changing, and resumes when you enter a newline.
- mname** Prefix the macro file */usr/lib/tmac/tmac.name* to the input files.
- raN** Set register *a* (one-character) to *N*.
- i** Read standard input after the input files are exhausted.
- q** Invoke the simultaneous input-output mode of the **rd** request.
- Tname** Prepare output for the specified terminal. Known *names* include **37** for the (default) Teletype Corporation Model 37 terminal, **tn300** for the GE TerminiNet 300 (or any terminal without half-line capability), **300S** for the DASI-300S, **300** for the DASI-300, and **450** for the DASI-450 (Diablo Hyterm).
- e** Produce equally-spaced words in adjusted lines, using full terminal resolution.
- h** Use output tabs during horizontal spacing to speed output and reduce output character count. **nroff** assumes tab settings to be every eight nominal character widths.

FILES

/tmp/ta*	Temporary file
/usr/lib/tmac/tmac.*	Standard macro files
/usr/lib/term/*	Terminal driving tables for nroff

SEE ALSO

troff(1), **eqn**(1), **tbl**(1), **ms**(7), **me**(7), **man**(7), **col**(1);
Unix Text Processing

NAME

obj2coff – convert OBJ format modules to COFF format modules

SYNOPSIS

obj2coff [-A *cpu,a88k*] *objmodule coffmodule*

DESCRIPTION

The **obj2coff** command converts OBJ files created under SR9.5 or later versions of Domain/OS to COFF files. **obj2coff** can convert individual modules or complete bound programs.

OBJ is the format used by all object files produced before SR10. Object files produced under SR10 and later versions of Domain/OS are in the COFF format. OBJ files will run under SR10, but many Apollo tools, including the linkers **bind** and **ld**, accept only COFF files as input.

With the -A *cpu,a88k* option, **obj2coff** marks the COFF output file as executable on Series 10000 workstations. Use this option only on data modules, not on modules containing code. **obj2coff** will not convert MC680x0 instructions to PRISM instructions.

The **obj2coff** command cannot convert object module libraries. See **lbr2ar**(1) for that purpose.

WARNINGS

If **obj2coff** encounters an object module stamped with an SR8 systype (*sys3*, *bsd4.1*, or any SR8), it converts the module to COFF and issues the following warning:

```
module is stamped with obsolete systype 'systype_name'
```

The *.sri* section of the COFF file will specify the obsolete systype.

obj2coff uses System V Release 3 and 4.3BSD semantics for all UNIX system calls. Some UNIX system calls may behave differently between Sys3 and SysV or between 4.1BSD and 4.3BSD, however. If you are converting a Sys3 or 4.1BSD object module to COFF, examine your source code carefully to avoid errors due to differences in call semantics.

OBJ files use stream 2 for error input and stream 3 for error output. COFF files use stream 2 for error output and have no automatically assigned stream for error input. A file converted from OBJ to COFF, therefore, may fail in an attempt to read error input and may generate the following error in writing error output:

```
operation attempted on unopened stream
```

SEE ALSO

lbr2ar(1)

NAME

od – octal, decimal, hex, ASCII dump

SYNOPSIS

od [*format*] [*file*] [[+]*offset*.[*b*] [*label*]]

DESCRIPTION

od displays *file* (or the standard input) in one or more dump formats as selected by the first argument. If you do not specify the first argument, **-o** is the default. Dumping continues until end-of-file.

The *offset* argument specifies the byte offset into the file where dumping is to begin. By default this argument is interpreted in octal. If you append a period (.) to the argument, **od** interprets *offset* in decimal; if *offset* begins with **x** or **0x**, **od** interprets it in hexadecimal. If you append **b** (or **B**), **od** interprets *offset* as a block count, where a block is 512 (or 1024) bytes. If you omit the *file* argument, you must prefix a plus sign (+) to the *offset* argument.

The radix of the displayed address is the same as the radix of the *offset*, if specified; otherwise it is octal.

od interprets the *label* argument as a pseudo-address for the first byte displayed. Shown in parentheses following the file offset, *label* is intended for use with core images to indicate the real memory address. The syntax for *label* is identical to that for *offset*.

OPTIONS

The following are the format option characters. An uppercase format character implies the long or double-precision form of the object.

- a[*pP*]** Interpret bytes as characters and display them with their ASCII names. If you also specify the **p** character, **od** underlines bytes with even parity. The **P** character causes bytes with odd parity to be underlined. Otherwise **od** ignores the parity bit.
- b** Interpret bytes as unsigned octal.
- c** Interpret bytes as ASCII characters. Certain non-graphic characters appear as C escapes: null=**\0**, backspace=**\b**, formfeed=**\f**, newline=**\n**, return=**\r**, tab=**\t**; others appear as three-digit octal numbers. Bytes with the parity bit set are displayed in octal.
- d** Interpret (short) words as unsigned decimal.
- f** Interpret long words as floating point.
- h** Interpret (short) words as unsigned hexadecimal.
- i** Interpret (short) words as signed decimal.
- l** Interpret long words as signed decimal.
- o** Interpret (short) words as unsigned octal.

- s[n]** Look for strings of ASCII graphic characters, terminated with a null byte. The value *n* specifies the minimum length string to be recognized. By default, the minimum length is three characters.
- v** Show all data. By default, **od** does not show display lines that are identical to the last line shown, but indicates them with an asterisk (*) in column 1.
- w[n]** The number of input bytes to be interpreted and displayed on each output line. If you do not specify **-w**, **od** reads 16 bytes for each display line. If you do not specify *n*, it defaults to 32.
- x** Interpret (short) words as hexadecimal.

SEE ALSO

strings(1)

BUGS

A filename argument can't start with a plus sign (+).

A hexadecimal offset can't be a block count.

You can specify only one filename argument.

NAME

pagesize – print system page size

SYNOPSIS

pagesize

DESCRIPTION

pagesize prints the size of a page of memory in bytes, as returned by **getpagesize(2)**. This program is useful in constructing portable shell scripts.

SEE ALSO

getpagesize(2)

NAME

pax – portable archive exchange

SYNOPSIS

pax [-clmopuvy] [-f *archive*] [-s *replstr*] [-t *device*] [*pattern...*]

pax -r [-cimnopuvy] [-f *archive*] [-s *replstr*] [-t *device*] [*pattern...*]

pax -w [-adimuvy] [-b *blocking*] [-f *archive*] [-s *replstr*] [-t *device*] [-x *format*] [*pathname...*]

pax -rw [-ilmopuvy] [-s *replstr*] [*pathname...*] *directory*

DESCRIPTION

Pax reads and writes archive files which conform to the **Archive/Interchange File Format** specified in *IEEE Std. 1003.1-1988*. *Pax* can also read, but not write, a number of other file formats in addition to those specified in the **Archive/Interchange File Format** description. Support for these traditional file formats, such as V7 *tar* and System V binary *cpio* format archives, is provided for backward compatibility and to maximize portability.

Pax will also support traditional *cpio* and System V *tar* interfaces if invoked with the name "cpio" or "tar" respectively. See the *cpio(1)* or *tar(1)* manual pages for more details.

Combinations of the **-r** and **-w** command line arguments specify whether *pax* will read, write or list the contents of the specified archive, or move the specified files to another directory.

The command line arguments are:

-w writes the files and directories specified by *pathname* operands to the standard output together with the pathname and status information prescribed by the archive format used. A directory *pathname* operand refers to the files and (recursively) subdirectories of that directory. If no *pathname* operands are given, then the standard input is read to get a list of pathnames to copy, one pathname per line. In this case, only those pathnames appearing on the standard input are copied.

-r *Pax* reads an archive file from the standard input. Only files with names that match any of the *pattern* operands are selected for extraction. The selected files are conditionally created and copied relative to the current directory tree, subject to the options described below. By default, the owner and group of selected files will be that of the invoking process, and the permissions and modification times will be the same as those in the archive.

The supported archive formats are automatically detected on input. The default output format is *ustar*, but may be overridden by the **-x format** option described below.

-rw *Pax* reads the files and directories named in the *pathname* operands and copies them to the destination *directory*. A directory *pathname* operand refers to the files and (recursively) subdirectories of that directory. If no

pathname operands are given, the standard input is read to get a list of pathnames to copy, one *pathname* per line. In this case, only those pathnames appearing on the standard input are copied. The directory named by the *directory* operand must exist and have the proper permissions before the copy can occur.

If neither the *-r* or *-w* options are given, then *pax* will list the contents of the specified archive. In this mode, *pax* lists normal files one per line, hard link pathnames as

pathname == *linkname*

and symbolic link pathnames (if supported by the implementation) as

pathname -> *linkname*

where *pathname* is the name of the file being extracted, and *linkname* is the name of a file which appeared earlier in the archive.

If the *-v* option is specified, then *pax* list normal pathnames in the same format used by the *ls* utility with the *-l* option. Hard links are shown as

<*ls -l listing*> == *linkname*

and symbolic links (if supported) are shown as

<*ls -l listing*> -> *linkname*

Pax is capable of reading and writing archives which span multiple physical volumes. Upon detecting an end of medium on an archive which is not yet completed, *pax* will prompt the user for the next volume of the archive and will allow the user to specify the location of the next volume.

Options

The following options are available:

- a* The files specified by *pathname* are appended to the specified archive.
- b blocking* Block the output at *blocking* bytes per write to the archive file. A *k* suffix multiplies *blocking* by 1024, a *b* suffix multiplies *blocking* by 512 and a *m* suffix multiplies *blocking* by 1048576 (1 megabyte). If not specified, *blocking* is automatically determined on input and is ignored for *-rw*.
- c* Complement the match sense of the the *pattern* operands.
- d* Intermediate directories not explicitly listed in the archive are not created. This option is ignored unless the *-r* option is specified.

- f *archive*** The *archive* option specifies the pathname of the input or output archive, overriding the default of standard input for **-r** or standard output for **-w**.
- i** Interactively rename files. Substitutions specified by **-s** options (described below) are performed before requesting the new file name from the user. A file is skipped if an empty line is entered and *pax* exits with an exit status of 0 if EOF is encountered.
- l** Files are linked rather than copied when possible.
- m** File modification times are not retained.
- n** When **-r** is specified, but **-w** is not, the *pattern* arguments are treated as ordinary file names. Only the first occurrence of each of these files in the input archive is read. The *pax* utility exits with a zero exit status after all files in the list have been read. If one or more files in the list is not found, *pax* writes a diagnostic to standard error for each of the files and exits with a non-zero exit status. the file names are compared before any of the **-i**, **-s**, or **-y** options are applied.
- o** Restore file ownership as specified in the archive. The invoking process must have appropriate privileges to accomplish this.
- p** Preserve the access time of the input files after they have been copied.
- s *replstr*** File names are modified according to the substitution expression using the syntax of *ed(1)* as shown:

-s */old/new/[gp]*

Any non null character may be used as a delimiter (a / is used here as an example). Multiple **-s** expressions may be specified; the expressions are applied in the order specified terminating with the first successful substitution. The optional trailing **p** causes successful mappings to be listed on standard error. The optional trailing **g** causes the *old* expression to be replaced each time it occurs in the source string. Files that substitute to an empty string are ignored both on input and output.

- t *device*** The *device* option argument is an implementation-defined identifier that names the input or output archive device, overriding the default of standard input for **-r** and standard output for **-w**.
- u** Copy each file only if it is newer than a pre-existing file with the same name. This implies **-a**.
- v** List file names as they are encountered. Produces a verbose table of contents listing on the standard output when both **-r** and **-w** are omitted, otherwise the file names are printed to standard error as they are encountered in the archive.

- x *format*** Specifies the output archive *format*. The input format, which must be one of the following, is automatically determined when the **-r** option is used. The supported formats are:
 - cpio*** The extended *CPIO* interchange format specified in **Extended CPIO Format in IEEE Std. 1003.1-1988**.
 - ustar*** The extended *TAR* interchange format specified in **Extended TAR Format in IEEE Std. 1003.1-1988**. This is the default archive format.
 - y** Interactively prompt for the disposition of each file. Substitutions specified by **-s** options (described above) are performed before prompting the user for disposition. EOF or an input line starting with the character **q** caused *pax* to exit. Otherwise, an input line starting with anything other than **y** causes the file to be ignored. This option cannot be used in conjunction with the **-l** option.

Only the last of multiple **-f** or **-t** options take effect.

When writing to an archive, the standard input is used as a list of pathnames if no *pathname* operands are specified. The format is one pathname per line. Otherwise, the standard input is the archive file, which is formatted according to one of the specifications in **Archive/Interchange File format in IEEE Std. 1003.1-1988**, or some other implementation-defined format.

The user ID and group ID of the process, together with the appropriate privileges, affect the ability of *pax* to restore ownership and permissions attributes of the archived files. (See *format-reading utility* in **Archive/Interchange File Format in IEEE Std. 1003.1-1988**.)

The options **-a**, **-c**, **-d**, **-i**, **-l**, **-p**, **-t**, **-u**, and **-y** are provided for functional compatibility with the historical *cpio* and *tar* utilities. The option defaults were chosen based on the most common usage of these options, therefore, some of the options have meanings different than those of the historical commands.

Operands

The following operands are available:

- directory*** The destination directory pathname for copies when both the **-r** and **-w** options are specified. The directory must exist and be writable before the copy or and error results.
- pathname*** A file whose contents are used instead of the files named on the standard input. When a directory is named, all of its files and (recursively) subdirectories are copied as well.
- pattern*** A *pattern* is given in the standard shell pattern matching notation. The default if no *pattern* is specified is *****, which selects all files.

EXAMPLES

The following command

```
pax -w -f /dev/rmt0 .
```

copies the contents of the current directory to tape drive 0.

The commands

```
mkdir newdir
cd olddir
pax -rw . newdir
```

copies the contents of *olddir* to *newdir* .

The command

```
pax -r -s '/*usr/*,,,' -f pax.out
```

reads the archive **pax.out** with all files rooted in *"/usr"* in the archive extracted relative to the current directory.

FILES

/dev/tty used to prompt the user for information when the *-l* or *-y* options are specified.

SEE ALSO

cpio(1), *find(1)*, *tar(1)*, *cpio(5)*, *tar(5)*

DIAGNOSTICS

Pax will terminate immediately, without processing any additional files on the command line or in the archive.

EXIT CODES

Pax will exit with one of the following values:

- 0 All files in the archive were processed successfully.
- >0 *Pax* aborted due to errors encountered during operation.

BUGS

Special permissions may be required to copy or extract special files.

Device, user ID, and group ID numbers larger than 65535 cause additional header records to be output. These records are ignored by some historical version of *cpio(1)* and *tar(1)*.

The archive formats described in **Archive/Interchange File Format** have certain restrictions that have been carried over from historical usage. For example, there are restrictions on the length of pathnames stored in the archive.

When getting an "ls -l" style listing on *tar* format archives, link counts are listed as zero since the *ustar* archive format does not keep link count information.

COPYRIGHT

Copyright (c) 1989 Mark H. Colburn.
All rights reserved.

Redistribution and use in source and binary forms are permitted provided that the above copyright notice is duplicated in all such forms and that any documentation, advertising materials, and other materials related to such distribution and use acknowledge that the software was developed by Mark H. Colburn and sponsored by The USENIX Association.

THE SOFTWARE IS PROVIDED "AS IS" AND WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

AUTHOR

Mark H. Colburn
NAPS International
117 Mackubin Street, Suite 1
St. Paul, MN 55102
mark@jhereg.MN.ORG

Sponsored by **The USENIX Association** for public distribution.

NAME

plot – graphics filters

SYNOPSIS

plot [**-Tterminal**] [**-rresolution**] [*file...*]

DESCRIPTION

These commands read plotting instructions (see **plot(5)**) from the standard input or the specified *files*, and in general produce plotting instructions suitable for a particular *terminal* on the standard output. The **-r** flag may be used to specify the device's output resolution (currently only the IMAGEN laser printer understands this option).

If run from a Display Manager window, a new window is created to display the plot.

If no *terminal* type is specified, the environment parameter \$TERM (see **environ(7)**) is used. Known *terminals* include:

apollo	Any Apollo display.
apollo.15P	Apollo 15-inch display.
apollo.19L	Apollo 19-inch display.
apollo.color	Apollo color display.
apollo.800.color	Apollo 800-line color display.
apollo.800.color1	Apollo 15-inch color display.
apollo.15.color1	Apollo 15-inch color display.
apollo.1280.color	Apollo 1280-line display.
4013	Tektronix* 4013 storage scope.
4014 or tek	Tektronix 4014 or 4015 storage scope with Enhanced Graphics Module. (Use 4013 for Tektronix 4014 or 4015 without the Enhanced Graphics Module).
450	DASI Hyterm 450 terminal (Diablo mechanism).
300	DASI 300 or GSI terminal (Diablo mechanism).
300S	DASI 300S terminal (Diablo mechanism).
aed	AED 512 color graphics terminal.
bitgraph or bg	BBN bitgraph graphics terminal.
imagen or lp	IMAGEN laser printer (default 240 dots-per-inch resolution).
crt	Any CRT terminal capable of running vi(1) .
dumb	Dumb terminals without cursor addressing or line printers.
vt125 or ggl	DEC vt125 terminal.
hp2648a or hp2648 or hp	Hewlett Packard 2648 graphics terminal.

hp7221 or **hp7**
ver
var

Hewlett Packard 7221 graphics terminal.
Versatec D1200A printer-plotter (not supported in BSD).
Benson Varian printer-plotter.
These versions of **plot** use the **-g** option of **lpr(1)** to send the result directly to the plotter device rather than to the standard output.

FILES

- /usr/bin/t4013**
- /usr/bin/tek**
- /usr/bin/tgmr**
- /usr/bin/t450**
- /usr/bin/t300**
- /usr/bin/t300s**
- /usr/bin/aedplot**
- /usr/bin/bgplot**
- /usr/bin/crtplot**
- /usr/bin/dumbplot**
- /usr/bin/gigipplot**
- /usr/bin/hpplot**
- /usr/bin/implot**
- /usr/ucb/lpr**

SEE ALSO

plot(3X), **plot(3F)**, **plot(5)**, **lpr(1)**

NAME

pr – print file

SYNOPSIS

pr [*options*] [*file* ...]

DESCRIPTION

pr produces a printed listing of one or more *files*. It separates output into pages headed by a date, the name of the file or a specified header, and the page number. If there are no file arguments, **pr** prints its standard input.

OPTIONS

Options apply to all following files, but may be reset between files.

- n** Produce *n*-column output.
- +n** Begin printing with page *n*.
- h** Take the next argument as a page header.
- wn** For purposes of multicolumn output, take the width of the page to be *n* characters instead of the default 72.
- f** Use formfeeds instead of newlines to separate pages. A formfeed is assumed to use up two blank lines at the top of a page. (Thus this option does not affect the effective page length.)
- ln** Take the length of the page to be *n* lines instead of the default 66.
- t** Do not print the five-line header or the five-line trailer normally supplied for each page.
- sc** Separate columns by the single character *c* instead of by the appropriate amount of white space. A missing *c* is interpreted as a tab.
- m** Print all *files* simultaneously, each in one column.

FILES

/dev/tty* to suspend messages.

DIAGNOSTICS

There are no diagnostics when **pr** is printing on a terminal.

BUGS

Interterminal messages via **write(1)** are forbidden during a **pr**.

SEE ALSO

cat(1), **print(1)**

NAME

prf – queue a file for printing by Domain/OS Aegis print spooler

SYNOPSIS

prf [*options*] *pathname*...

DESCRIPTION

The **prf** command queues a file for printing. After successfully queuing a file, **prf** displays a message containing the full pathname of the file that you queued.

You can execute **prf** once for each file that you want to print (specifying all the necessary options every time), or you can enter **prf**'s interactive mode and hand files to the program continuously. See the examples for a sample interactive session.

Files queued by **prf** are physically printed by **prsvr**, the print server, running as a background task under control of **prmgr**, the print manager.

When you invoke **prf**, it first sets all options to their default states. Next, it looks for the print options file called **user_data/startup.prf** unless you invoke **prf** with the **-ndb** option. If **prf** locates the option file, it executes the options contained in the file to configure the current session. Finally, it processes all options on the command line.

pathname (optional) Specify the file to be printed. Multiple pathnames and pathname wildcarding are permitted.

Default if omitted: read standard input

OPTIONS

The following options can appear on the shell command line or in **prf** interactive mode. In addition, you can place one or more options in a **prf** option file so that they are executed automatically whenever you invoke **prf**.

Many of the options have default values that are specified in the **prsvr** configuration file established for each printer in the network by the system administrator. If you omit these options, your file is printed using the values specified in the **prsvr** configuration file. For example, omission of the **-banner** option could cause your file to be printed with a banner page if the **prsvr** configuration file specifies one.

Not all options apply to all printers. Refer to Appendix A of *Printing in the Aegis Environment* for a printer option matrix.

Options Applying to All File Types

-dia[logue]

Invokes the menu-based version of the **prf** command. This interactive version runs a graphics interface inside the process window in which **prf -dia** was invoked. You control the operation of **prf -dia** by pointing the cursor at a desired menu item and pressing M1 (if you have a mouse) or F1 or the space bar if you do not have a mouse. Internal help is available by pointing at a menu item and pressing the M3 (if you have a mouse) or the <HELP> key R6S if you do not have a mouse.

prf -dia requires no arguments or options on the shell command line. To exit **prf -dia** and return to the shell, select the "quit" menu item.

prf -dia performs exactly the same functions as the command-line based **prf**.

-inter[active]

Enter interactive mode.

-sea[rch_dir] [on|off]

Searches through all the directories of all the active processes on your node for the file(s) to be printed. This option is most useful in interactive mode, when the working directory of the **prf** process may be different from the working directory of the file to be printed. If this option is specified without **on** or **off**, **on** is assumed.

If the option is not specified, the default is **off**.

-cop[ies] n

Prints multiple copies of the file, where *n* is the requested number of copies. If **-cop[ies]** is specified, *n* is required. The default is one copy.

-pr[inter]name

Specifies the name of the printer that should print the file. This option is useful only if more than one printer is in use on the network, or if a printer has been assigned a nonstandard name with the **printer_name** configuration directive in the **prsvr** command. If you omit this option, **prf** uses the default printer name, **p**. Note that **p** is also the default printer name used by the print server.

-spool[_node] spool_node_name

Use this option only if you are queuing jobs to a pre-SR10 print server connected to a spool directory (**/sys/print**) that is different from the one specified by your node. By default, SR10 printers find the spool node for you.

-nc[opy]

Prints the specified file from its location in the user-specified directory, bypassing **/sys/print/spooler**. If you select this option, **prf** defaults to the no-delete (**-nd**) option. If you specify the delete (**-d**) option, the file is deleted at the completion of the print request. If you use this option (with or without the delete option), do not open and alter the print file before the print job is completed.

-d[ele]te] (default)

Deletes the print file at the completion of the print job.

-nd[ele]te]

Does not delete the print file when the print server is finished printing it. This becomes default if **-nc** is specified.

- user[*username*]** Specifies the user name that appears on the banner page of the printed file. The alarm facility of **prf** also uses this name to determine who should be notified when printing is complete (see **-sig** below). This means that this name must be a valid log-in name (unless you don't care about sending an alarm).
- The default is the current log-in name.
- sig[*nal*] {*alarm*|*off*}** Requests an alarm server signal when the file has finished printing.
- The default is **off**.
- ban[*ner*] [*on*|*off*]** Enables/disables banner page. If this option is specified without **on** or **off**, **on** is assumed. However, if the banner setting in the **prsvr** configuration file is **off**, no banner is printed.
- If the option is not specified, the default is **on**.
- config[_*file*] [*pathname*]** Specifies a file containing further **prf** options, one per line. Do not use prefixed hyphens (-) with the option names in the configuration file. If *pathname* is omitted, **prf** executes the **prf** option file `~/user_data/startup.prf`.
- ndb** Suppresses processing of the **prf** option file.
- trans[*parent*] [*on*|*off*]** Specifies that the file being printed is passed directly to the printer driver routine with no processing by the print server.
- If this option is specified without **on** or **off**, **on** is assumed. If the option is not specified, the default is **off**.
- filter *string*** Specifies a filter string that will be used by the print server to process the job. This option overrides the default processing done by the print server. It is most often used to invoke filters that have been added to the print server. The format of the string is "filter1 : filter2", where filter1 and filter2 are composed of strings of the form "type1\$type2" and "type2\$type3". Note that the output type of filter *n* must equal the input type of filter *n*+1.
- paper_size {*a*|*b*|*legal*|*a3*|*a4*|*a5*|*b4*|*b5*}** Selects the paper size. You must specify one of the following size codes:

Code	Size in inches (mm)
a	8.50 x 11.00
b	11.00 x 17.00
legal	8.50 x 14.00
a3	11.69 x 16.54 (297mm x 420mm)
a4	8.27 x 11.69 (210mm x 297mm)
a5	5.38 x 8.27 (137mm x 210mm)
b4	9.84 x 13.90 (257mm x 364mm)
b5	5.93 x 9.89 (182mm x 257mm)

This option is available only for printers that have multiple print trays. Because **prf** assumes that the correct paper is in the printer's paper tray, you should check the paper tray before printing. The default paper size is specified in the **prsvr** configuration file.

-orient[ation] {port[rait]||land[scape]}

Selects the page orientation. **portrait** specifies that the text or x-axis of the bitmap is printed parallel to the short edge of the paper. **landscape** specifies that the text or x-axis of the bitmap is printed parallel to the long edge of the paper and perpendicular to the short leading edge.

The default is **portrait**.

Options Applying to Text Files Only

-margins [on|off]

Enables/disables margins. If this option is specified without **on** or **off**, **on** is assumed.

If the option is not specified, the default is **on**.

-top *n*

Specifies top page margin, in inches. The default is a value specified in the **prsvr** configuration file.

-bot[tom] *n*

Specifies bottom page margin, in inches. The default is a value specified in the **prsvr** configuration file.

-right *n*

Specifies right margin, in inches. The default is 0 inches.

-left *n*

Specifies left margin, in inches. The default is 0 inches.

-headers [on|off]

Enables/disables page headers and footers. If this option is specified without **on** or **off**, **on** is assumed.

If the option is not specified, the default is **off**.

-head[_string] *l-string/c-string/r-string*

Specifies contents of left, center, and right components

of the page header generated by **prf**. Components can be empty strings. The following special characters return the values indicated when they appear in the header strings:

Character	Return Value
@	Escape character
#	Current Page number with 1 leading and 1 trailing space
%	Current date
!	Filename
&	Filename's last time, date modified
*	Insert a space in text string (literal spaces are not allowed)

Example: **-head !/Page#/%** produces a header with the filename in the left component, the string "Page" followed by the current page number in the center component, and the current date in the right component. The default header is a string specified in the **prsvr** configuration file.

-foot[_string] *l-string/c-string/r-string*

Specifies contents of page footers. The format is the same as for **-head** above.

-ftn [on|off]

Enables/disables FORTRAN carriage control. **-ftn on** causes the print server to use FORTRAN forms control even if the file does not have the FORTRAN carriage-control flag. Use of this option causes **prf** to interpret the first character of each line as a FORTRAN carriage control character (and not print it). This can be unfortunate if the file has ASCII carriage control, so be careful. **-ftn off** causes the print server to print the contents of column one rather than trying to interpret it as FORTRAN forms control. If this option is specified without **on** or **off**, **on** is assumed.

If the option is not specified, the default is **off**.

-wrap [on|off]

Enables/disables automatic line wrapping. When enabled, **prf** wraps lines that exceed the right margin. When disabled, **prf** truncates lines that exceed the right margin. If this option is specified without **on** or **off**, **on** is assumed.

If the option is not specified, the default is **off**.

-col[umns] {1 2}	Specifies single-or double-column printing. The default state is single column.
-lpi <i>n</i>	Specifies the line-spacing factor. <i>n</i> is an integer indicating the number of lines per inch. The default value is dependant on the point size of the font.
Options for Variable Font and Pitch	
-pitch <i>n</i>	Sets the printer pitch (characters/inch).
-point <i>n</i>	Sets the point size for the font to be used. This is a real number that specifies size in points. A point equals 1/72 inch.
-weight {light medium bold}	Sets the weight of the font to be used. The default is medium .
-lq [on off]	Specifies that the document is to be printed in letter quality (on) or in draft (off) mode. If this option is specified without on or off , on is assumed. If the option is not specified, the default is off (draft mode).
Options Applying to Plot Files	
-res[olution] <i>n</i>	Specifies output plot resolution in dots per inch. If you specify a resolution not available on the particular printer, prsvr prints the file at the closest available resolution. The default resolution is specified in the prsvr configuration file.
-white[_space] <i>n</i>	Specifies the amount of white space (in inches) to appear between multiple plots in one file. The default is three inches.
-bw[_rev] [on off]	Enables/disables black and white reversal for bitmaps. If this option is specified without on or off , on is assumed. If the option is not specified, the default is off .
-magn[ification] <i>n</i>	Specifies bitmap magnification value. <i>n</i> is an integer in the range -1 to 16. The values have the following meanings:

- 1 Selects auto-scaling to magnify the bitmap to fill the available page space.
- 0 Selects one-to-one scaling between the display and the printer for GMF bitmaps. (For GPR bitmaps, this translates to magnification 1.)
- 1-16 Selects the magnification indicated by value. Where 1 equals 1-to-1, 2 equals 2x, etc.

Default if omitted: *n* is 0

Information Request Options

- check [-pr *printer_name*]** Checks for the existence of the specified printer. If the printer does not exist or is unavailable, an error message is returned.
- list_pr[inters] [*print_manager_name*]** Lists the names and status of all printers currently attached to the network or to the specified print manager.
- list_s[ites]** Lists the names of all print managers currently in the network.
- sig_pr[inter] *printer_name* {-abort|-sus[pend]|cont[inue]|check}** Signals the printer to abort, suspend, or continue an active print job. **check** tests for the existence of an active print server and forces the server to re-register with the print manager. If the print server does not exist or is unavailable, an error message is returned.
- pre10** Allows you to queue print requests to a pre-SR10 print server.
- r[ead] [*printer*]** List queue entries for the specified printer. If *printer* is omitted, the contents of the queue (determined by the current setting of **-pr**) are listed.
- can[cel] *job_id*** Cancel printing of the specified file. Note that you must specify the job ID assigned by **prmgr** when the file is queued. Use the **read** command to display the names and job IDs of currently queued files. This command affects jobs in the print queue; it does not cancel a job being printed. To halt a job being printed, use **-sig_pr** with abort specified.

COMMANDS

Once **prf** has been invoked in interactive mode (see **-inter** above), it accepts the following interactive commands at the "prf> " prompt (in addition to the options already discussed).

p[rint] [<i>print_file_pathname</i>] [<i>options</i>]	Queue the specified file for printing.
q[uit]	Quit interactive mode and return to the shell.
sh[ell]	Create a shell command line. This command allows you to issue shell commands without leaving prf interactive mode. When you finish entering shell commands, type CTRL/Z. This returns you to prf interactive mode. Your previous prf option settings remain undisturbed by the intervening shell commands.
init[ialize]	Reset prf parameters to their default values.
wd [<i>pathname</i>]	Execute the shell command wd (<i>working_directory</i>) to set or display the working directory.
get option	Display the value of the prf option specified. Use this command to show the settings of the various prf parameters.

EXAMPLES

The following example, queues the file named **mary** for printing and forces FORTRAN carriage returns:

```
$ prf mary -ftn -pr my_printer
"//node1/my_dir/mary" queued to printer my_printer.
$
```

This example shows the types of commands that might appear in the default **prf** configuration file **/user_data/startup.prf**:

```
pr ge
foot %/my_file/&
sig alarm
left 0.25
```

The following example shows a sample interactive session:

```
$ prf -inter
prf> get pr
pr = p
prf> -pr cx
prf> get pr
pr = cx
prf> -pitch 20
prf> print test_file.pas
"//node1/my_dir/test_file.pas" queued for printing.
prf> q
$
```

To run **prf** interactively in a process devoted to it, insert the following command in the start-up file that you use to start the DM:

```
cp -i -c 'P' /com/prf -inter -n print_file
```

The above command creates a **prf** process and turns its window into an icon using the print icon character in (**/sys/dm/fonts/icons**). Issue the DM command **icon** to change the icon window into its full-size format.

NAME

print – **pr** to the line printer

SYNOPSIS

print *files*

DESCRIPTION

print uses **pr** to print a copy of each named *file* on the line printer. **print** is a one-line shell script:

lpr -p \$*

SEE ALSO

lpr(1), **pr**(1)

NAME

printenv – print out the environment

SYNOPSIS

printenv [*name*]

DESCRIPTION

printenv prints out the values of the variables in the environment. If a *name* is specified, only its value is printed.

If a *name* is specified and it is not defined in the environment, **printenv** returns exit status 1; otherwise it returns status 0.

SEE ALSO

sh(1), environ(7), csh(1)

NAME

prism – get processor type truth value for prism

SYNOPSIS

prism

DESCRIPTION

prism will return a true value (exit code of 0) if you are on a Prism processor. Otherwise, it will return a false (non-zero) value. This command is particularly useful within makefiles (see *make*(1)) and shell procedures (see *sh*(1)) to increase portability.

SEE ALSO

sh(1), test(1), true(1), make(1).

NAME

prof – display profile data

SYNOPSIS

prof [**-alnzs**] [**-v** [**-low** [**-high**]]] [*objfile* [*profile* ...]]

DESCRIPTION

prof interprets the file produced by the **monitor** subroutine. In normal operation, **prof** reads the symbol table in the object file *objfile* and correlates it with the profile file *profile*. **prof** uses the default files **a.out** (object file) and **mon.out** (profile file) if you do not give explicit filenames.

For each external symbol, the percentage of time spent executing between that symbol and the next is printed (in decreasing order), together with the number of times that routine was called and the number of milliseconds per call. If you specify more than one profile file, the output represents the sum of the profiles.

In order for the number of calls to a routine to be tallied, the **-p** option of **cc(1)** or **f77(1)** must have been given when the file containing the routine was compiled. This option also arranges for the profile file to be produced automatically.

OPTIONS

- a** Report all symbols rather than only external symbols.
- l** Sort output by symbol value.
- n** Sort output by number of calls.
- z** Print routines which have zero usage (as indicated by call counts and accumulated time).
- s** Produce a summary of profile file in **mon.sum** (Not supported in Domain/OS BSD).
- v** Suppress printing and produce a graphic version of the profile on the standard output for display by the **plot(1)** filters (**-v** is not supported in Domain/OS BSD).

FILES

mon.out for profile
a.out for namelist

BUGS

Beware of quantization errors.

prof is confused by **f77**, which puts the entry points at the bottom of subroutines and functions.

SEE ALSO

monitor(3), **profil(2)**, **cc(1)**, **plot(1G)**

NAME

prs - print an SCCS file

SYNOPSIS

prs [**-d**[*dataspec*]] [**-r**[*SID*]] [**-e**] [**-l**] [**-c**[*date-time*]] [**-a**] *file* ...

DESCRIPTION

prs prints, on the standard output, part or all of an SCCS (Source Code Control System) file in a user-supplied format. If you name a directory, **prs** behaves as though each file in the directory is specified as a named file, except that it silently ignores non-SCCS and unreadable files. If a dash (-) is given in place of a filename, **prs** reads the standard input, taking each line to be the name of an SCCS file or directory to be processed. Options to **prs** may appear in any order. Each argument applies independently to each named file.

OPTIONS

- d**[*dataspec*] Specify the output data specification. The *dataspec* is a string consisting of SCCS file data keywords interspersed with optional user-supplied text.
- r**[*SID*] Specify the SCCS Identification (SID) string of the delta for which information is desired. If you do not specify an *SID*, then **prs** assumes the *SID* to be that of the most recently created delta.
- e** Request information for all deltas created earlier than and including the delta designated via the **-r** keyletter or the date given by the **-c** option.
- l** Request information for all deltas created later than and including the delta designated via the **-r** keyletter or the date given by the **-c** option.
- c**[*date-time*] Specify *date-time* as cutoff for requesting information. This cutoff *date-time* appears in the following form:

YY[MM[DD[HH[MM[SS]]]]]

Units omitted from the *date-time* default to their maximum possible values; that is, **-c8502** is equivalent to **-c850228235959**. Any number of non-numeric characters may separate the various two-digit pieces of the cutoff date in the following form:
-c85/2/2 9:22:25.
- a** Request printing of information for both removed (delta type = **R**) and existing (delta type = **D**) deltas. Refer to **rmDEL(1)** for more information. If you do not specify the **-a** keyletter, **prs** provides information only on existing deltas.

DATA KEYWORDS

Data keywords specify those parts of an SCCS file to be retrieved and output. All parts

of an SCCS file have an associated data keyword. Refer to `sccsfile(4)` for more information about the structure of these file types. There is no limit on the number of times a data keyword may appear in a *dataspec*.

`prs` prints the user-supplied text, and appropriate values (extracted from the SCCS file) substituted for the recognized data keywords in the order of appearance in the *dataspec*. The format of a data keyword value is either "Simple" (S), in which keyword substitution is direct, or "Multiline" (M), in which keyword substitution is followed by a carriage return. User-supplied text is any text other than recognized data keywords.

A tab is specified by `\t`, and a carriage return/newline is specified by `\n`. The default data keywords are: `":Dt:\n:DL:\nMRS:\n:MR:COMMENTS:\n:C:"`

SCCS Files Data Keywords				
Keyword	Data Item	File Section	Value	Format
:Dt:	Delta information	Delta Table	See below*	S
:DL:	Delta line statistics	"	:Li:/Ld:/Lu:	S
:Li:	Lines inserted by Delta	"	nnnnn	S
:Ld:	Lines deleted by Delta	"	nnnnn	S
:Lu:	Lines unchanged by Delta	"	nnnnn	S
:DT:	Delta type	"	D or R	S
:I:	SCCS ID string (SID)	"	:R::L::B::S:	S
:R:	Release number	"	nnnn	S
:L:	Level number	"	nnnn	S
:B:	Branch number	"	nnnn	S
:S:	Sequence number	"	nnnn	S
:D:	Date Delta created	"	:Dy:/Dm:/Dd:	S
:Dy:	Year Delta created	"	nn	S
:Dm:	Month Delta created	"	nn	S
:Dd:	Day Delta created	"	nn	S
:T:	Time Delta created	"	:Th::Tm::Ts:	S
:Th:	Hour Delta created	"	nn	S
:m:	Minutes Delta created	"	nn	S
:Ts:	Seconds Delta created	"	nn	S
:P:	Programmer who created Delta	"	logname	S
:DS:	Delta sequence number	"	nnnn	S
:DP:	Predecessor Delta seq-no.	"	nnnn	S
:DI:	Seq-no. of deltas incl., excl., ignored	"	:Dn:/Dx:/Dg:	S
:Dn:	Deltas included (seq #)	"	:DS: :DS:...	S
:Dx:	Deltas excluded (seq #)	"	:DS: :DS:...	S

SCCS Files Data Keywords (Contd.)

Keyword	Data Item	File Section	Value	Format
:Dg:	Deltas ignored (seq #)	"	:DS: :DS:...	S
:MR:	MR numbers for delta	"	text	M
:C:	Comments for delta	"	text	M
:UN:	User names	User Names	text	M
:FL:	Flag list	Flags	text	M
:Y:	Module type flag	"	text	S
:MF:	MR validation flag	"	yes or no	S
:MP:	MR validation pgm name	"	text	S
:KF:	Keyword error/warning flag	"	yes or no	S
:BF:	Branch flag	"	yes or no	S
:J:	Joint edit flag	"	yes or no	S
:LK:	Locked releases	"	:R:...	S
:Q:	User defined keyword	"	text	S
:M:	Module name	"	text	S
:FB:	Floor boundary	"	:R:	S
:CB:	Ceiling boundary	"	:R:	S
:Ds:	Default SID	"	:I:	S
:ND:	Null delta flag	"	yes or no	S
:FD:	File descriptive text	Comments	text	M
:BD:	Body	Body	text	M
:GB:	Gotten body	"	text	M
:W:	A form of what(1) string	N/A	:Z::M::I:	S
:A:	A form of what(1) string	N/A	:Z::Y: :M: :I::Z:	S
:Z:	what(1) string delimiter	N/A	@(#)	S
:F:	SCCS file name	N/A	text	S
:PN:	SCCS file path name	N/A	text	S

* :Dt: = :DT: :I: :D: :T: :P: :DS: :DP:

EXAMPLES

```
% prs -d "Users and/or user IDs for :F: are:0UN:" s.file
```

Users and/or user ids for s.file are:

xyz

131

abc

```
% prs -d "Newest delta for pgm :M:: :I: Created :D: By :P:" -r s.file
```

Newest delta for pgm main.c: 3.7 Created 85/12/1 By cas

A simple command line without options, such as **prs s.file**, may produce the following on the standard output, for each delta table entry of the "D" type:

```
D 1.1 85/12/1 00:00:00 cas 1 000000/000000/000000
MRS:
b178-12345
b179-54321
COMMENTS:
this is the comment line for s.file initial delta
%
```

FILES

/tmp/pr????

DIAGNOSTICS

Use **help(1)** for explanations.

SEE ALSO

admin(1), **delta(1)**, **get(1)**, **help(1)**, **sccs(1)**, **sccsfile(4)**;
Using Your BSD Environment.

NAME

ps – process status

SYNOPSIS

ps [*aceglnstuvwxU#* [*node_spec*]]

DESCRIPTION

ps prints information about processes. Normally, only your processes are candidates to be printed by **ps**. Specifying the **a** option causes other users' processes to be candidates to be printed. Specifying the **x** option includes processes without control terminals in the candidate pool.

All output formats include, for each process, the process ID (PID), control terminal of the process (TT), CPU time used by the process (TIME) including both user and system time, the state of the process (STAT), and an indication of what command is running (COMMAND).

The state is given by a sequence of four letters, e.g. RWNA. The first letter indicates the runnability of the process: R for runnable processes, T for stopped processes, P for processes in page wait, D for those in disk (or other short term) waits, S for those sleeping for less than about 20 seconds, and I for idle (sleeping longer than about 20 seconds) processes. The second letter indicates whether a process is swapped out, showing W if it is, or a blank if it is loaded. A process which has specified a soft limit on memory requirements and which is exceeding that limit shows a greater-than character (>); such a process is (necessarily) not swapped. The third letter indicates whether a process is running with altered CPU scheduling priority, as under **nice**(1). If the process priority is reduced, an N is shown; if the process priority has been artificially raised, then a less-than character (<) is shown. Processes running without special treatment have just a blank.

OPTIONS

- | | |
|----------|---|
| a | Request information about all processes with terminals (ordinarily only your own processes are displayed). |
| c | Print the command name, as stored internally in the system for purposes of accounting, rather than the command arguments, which are kept in the process's address space. This is more reliable, if less informative, since the process is free to destroy the latter information. |
| e | Print the environment as well as the arguments to the command. |
| g | Ask for all processes. Without this option, ps prints only "interesting" processes. Processes are deemed to be uninteresting if they are process group leaders. This normally eliminates top-level command interpreters and processes waiting for users to log in on free terminals. |
| k | Not supported in Domain/OS BSD. |
| l | Ask for a long listing, with fields PPID, CP, PRI, NI, ADDR, SIZE, RSS and WCHAN as described below. |

n	Produce numerical output. In a long listing, the WCHAN field is printed numerically rather than symbolically, or, in a user listing, the USER field is replaced by a UID field.
s	Add the size SSIZ of the kernel stack of each process (for use by system maintainers) to the basic output format.
tx	Restrict output to processes whose controlling tty is <i>x</i> (which should be specified as printed by ps , for example, t3 for tty3, tco for console, td0 for ttyd0, t? for processes with no tty, t for processes at the current tty, etc). This must be the last option given.
u	Produce a user oriented output, including the fields USER, %CPU, NICE, SIZE, and RSS as described below.
v	Produce a version of the output containing virtual memory statistics, including the fields RE, SL, PAGEIN, SIZE, RSS, LIM, TSIZ, TRS, %CPU and %MEM, described below.
w	Use a wide output format (132 columns rather than 80). If repeated, e.g., ww , use arbitrarily wide output. This information is used to decide how much of long commands to print.
x	Ask about all processes, even those with no terminal.
U	Update a private database where ps keeps system information. Thus, the command ps U should be included in the <i>/etc/rc</i> file. Domain/OS BSD does not support <i>/etc/psdatabase</i> .
#	Restrict the output to the process whose process number is specified (the process number is indicated here by the pound sign, #). This option must also be last.
<i>node_spec</i>	Show information about processes running on the specified node. Must be specified following at least one other option.

VARIABLE OUTPUT FIELDS

Fields that are not common to all output formats are described below.

USER	Name of the owner of the process
%CPU	CPU utilization of the process; this is a decaying average over up to a minute of previous (real) time. Since the time base over which this is computed varies (since processes may be very young), it is possible for the sum of all %CPU fields to exceed 100%.
NICE	(or NI) Process scheduling increment (see setpriority(2))
SIZE	Virtual size of the process (in 1024 byte units)
RSS	Real memory (resident set) size of the process (in 1024 byte units)
LIM	Soft limit on memory used, specified via a call to setrlimit(2) ; if no limit has been specified then shown as <i>xx</i>
TSIZ	Size of text (shared program) image
TRS	Size of resident (real memory) set of text

%MEM	Percentage of real memory used by this process.
RE	Residency time of the process (seconds in core)
SL	Sleep time of the process (seconds blocked)
PAGEIN	Number of disk I/O's resulting from references by the process to pages not loaded in core.
UID	Numerical user-ID of process owner
PPID	Numerical ID of parent of process
CP	Short-term CPU utilization factor (used in scheduling)
PRI	Process priority (non-positive when in non-interruptible wait)
ADDR	Swap address of the process
WCHAN	Event on which process is waiting (an address in the system). A symbol is chosen that classifies the address, unless numerical output is requested (see the n flag). In this case, the initial part of the address is trimmed off and is printed hexadecimally, e.g., 0x80004000 prints as 4000.

F Flags associated with process as in `<sys/proc.h>`:

SLOAD	000001	in core
SSYS	000002	swapper or pager process
SLOCK	000004	process being swapped out
SSWAP	000008	save area flag
STRC	000010	process is being traced
SWTED	000020	another tracing flag
SULOCK	000040	user settable lock in core
SPAGE	000080	process in page wait state
SKEEP	000100	another flag to prevent swap out
SDLYU	000200	delayed unlock of pages
SWEXIT	000400	working on exiting
SPHYSIO	000800	doing physical i/o (bio.c)
SVFORK	001000	process resulted from vfork()
SVFDONE	002000	another vfork flag
SNOVM	004000	no vm, parent in a vfork()
SPAGI	008000	init data space on demand from inode
SANOM	010000	system detected anomalous vm behavior
SUANOM	020000	user warned of anomalous vm behavior
STIMO	040000	timing out during sleep
SDETACH	080000	detached inherited by init
SOUSIG	100000	using old signal mechanism

NOTES

A process that has exited and has a parent that has not yet waited for the process is marked <defunct>. A process that is blocked trying to exit is marked <exiting>. While **ps** makes an educated guess as to the filename and arguments given when the process was created by examining memory or the swap area, this method is inherently somewhat unreliable. In any event a process is entitled to destroy this information, so the names cannot be counted on too much.

BUGS

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

FILES

/dev Searched to find tty names

SEE ALSO

kill(1), tb(1)

NAME

ptx – permuted index

SYNOPSIS

ptx [*options*] ... [*input* [*output*]]

DESCRIPTION

ptx generates a permuted index to file *input* on file *output* (standard input and output default). It has three phases: the first does the permutation, generating one line for each keyword in an input line and rotating the keyword to the front; the second sorts the permuted file; and the third rotates the sorted lines so the keyword comes at the middle of the page. **ptx** produces output in the form

.xx "tail" "before-keyword" "keyword-and-after" "head"

where *.xx* may be a user-defined **nroff**(1) or **troff**(1) macro for subsequent formatting. The *before-keyword* and *keyword-and-after* fields incorporate as much of the line as will fit around the keyword when it is printed at the middle of the page. If the entire input line is too wide to fit in these two fields, **ptx** inserts a slash (/) to indicate where text was omitted. The fields *tail* and *head*, at least one of which is an empty string, are wrapped-around pieces small enough to fit in the unused space at the opposite end of the line.

The permuted index for this manual was generated using **ptx**.

OPTIONS

- f** Fold uppercase and lowercase letters for sorting.
- t** Prepare the output for the phototypesetter. The default line length is 100 characters.
- w *n*** Use the next argument, *n*, as the width of the output line. The default line length is 72 characters.
- g *n*** Use the next argument, *n*, as the number of characters to allow for each gap among the four parts of the line as finally printed. The default gap is three characters.
- o *only*** Use as keywords only the words given in the *only* file.
- i *ignore*** Do not use as keywords any words given in the *ignore* file. If the **-i** and **-o** options are missing, use */usr/lib/elgn* as the *ignore* file.
- b *break*** Use the characters in the *break* file to separate words. In any case, **ptx** always uses tab, newline, and space characters as break characters.
- r** Take any leading nonblank characters of each input line to be a reference identifier (as to a page or chapter) separate from the text of the line. Attach that identifier as a fifth field on each output line.

BUGS

Line length counts do not account for overstriking or proportional spacing.

FILES

`/usr/bin/sort`
`/usr/lib/eign`

SEE ALSO

`nroff(1)`, `troff(1)`

NAME

pwd – working directory name

SYNOPSIS

pwd

DESCRIPTION

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

SEE ALSO

cd(1), **csh**(1), **getwd**(3)

BUGS

In **csh**(1) the command **dirs** is always faster (although it can give a different answer in the case that the current directory or a containing directory was moved after the shell descended into it, or when a symbolic link was traversed).

NAME

ranlib – convert archives to random libraries

SYNOPSIS

ranlib [**-t**] *archive* ...

DESCRIPTION

NOTE: **ranlib** is not necessary on Domain/OS BSD systems; however, it is provided for compatibility.

ranlib converts each *archive* to a form which the loader can load more rapidly. **ranlib** does this by adding a table of contents called **__SYMDEF** to the beginning of the archive. **ranlib** uses **ar**(1) to reconstruct the archive, so that sufficient temporary file space must be available in the file system which contains the current directory.

If given the **-t** option, **ranlib** only "touches" the archives and does not modify them. This is useful after copying an archive or using the **-t** option of **make**(1) in order to avoid having **ld**(1) complain about an "out of date" symbol table.

SEE ALSO

ld(1), **ar**(1), **lorder**(1), **make**(1)

BUGS

Because generation of a library by **ar** and randomization of the library by **ranlib** are separate processes, phase errors are possible. The loader, **ld**, warns when the modification date of a library is more recent than the creation date of its dictionary; but this means that you get the warning even if you only copy the library.

NAME

ratfor – rational FORTRAN dialect

SYNOPSIS

ratfor [*option* ...] [*filename* ...]

DESCRIPTION

ratfor converts a rational dialect of FORTRAN into ordinary irrational FORTRAN. **ratfor** provides control flow constructs essentially identical to those in C:

statement grouping:

```
{ statement; statement; statement }
```

decision-making:

```
if (condition) statement [ else statement ]
```

```
switch (integer value) {
```

```
    case integer:    statement
```

```
    ...
```

```
    [ default: ]    statement
```

```
}
```

loops: **while** (*condition*) *statement*

```
for (expression; condition; expression) statement
```

```
do limits statement
```

```
repeat statement [ until (condition) ]
```

```
break
```

```
next
```

and some syntactic sugar to make programs easier to read and write:

free form input:

multiple statements/line; automatic continuation

comments:

```
# this is a comment
```

translation of relationals:

>, >=, etc., become .GT., .GE., etc.

return (*expression*)

returns expression to caller from function

define: **define** *name replacement*

include: **include** *filename*

ratfor is best used with **f77(1)**.

SEE ALSO

f77(1)

Domain FORTRAN Language Reference

B. W. Kernighan and P. J. Plauger, *Software Tools*, Addison-Wesley, 1976.

NAME

rbak – restore or index a magnetic media backup file

SYNOPSIS

```
rbak {-f filename|-fid id} [-dev | m[unit] | f | ct]
      [-int|-index] [-sla|-nsla] [-anys] [-reo] [-pr pn]
      [-cr|-r|-ms|-md] [-force] [-du] [-l|-ld|-lf|-ll]
      [-reten|-nreten] [-rewind] [-dac|-sac]
      [-from filename] [-pdt] [-stdln] [{pn|-all}
      [-as disk_pn]}...
```

DESCRIPTION

rbak restores objects from the backup input media written by **wbak** (write_backup). The backup input media can be one of magnetic media, file or standard input.

Use **wbak** and **rbak** to back up disks and to transfer information between separate Domain installations. (Use the **rwmt** (read_write_magtape) command to transfer information to and from non-Domain installations.)

rbak operates in either index or interchange mode. To restore objects to disk, use interchange mode (**-int**). To list object names on standard output, without restoring any information to disk, use index mode (**-index**).

pn – pathname (optional) Specify name of the object to be indexed or restored to disk. This may be a directory, file, or link. If the object is being restored, the new disk object has the same name. If you wish the disk file to be saved under a different name, use **-as** (below). Multiple pathnames are permitted; however, wildcarding is not supported.

Default if omitted: must use **-all**

OPTIONS

Backup File Identifiers

One of the following options is required.

- | | |
|----------------------------|---|
| -f <i>file_no</i> | Reads the back up file with the file number specified. You assigned this number with wbak . |
| -f <i>cur</i> | Begins reading at current position on the back up medium. |
| -fid <i>file_id</i> | Reads the back up filename specified. You assigned this name using wbak . |
| -int (default) | Selects interchange mode. Backup files are restored to disk. |
| -index | Selects index mode. Backup filenames are listed on standard output; no information is restored to disk. |

Catalog Control

-all

Restores or indexes all the objects in the back up file specified. This option is required if you do not use the **pn** argument to indicate a particular object to be indexed or restored.

-as disk_pn

Restores the object specified and assign a different disk pathname *disk_pn*. This option is valid only when used with the **pn** argument on the **rbak** command line.

-cr (default)

Specifies create mode. **rbak** does not restore objects if their names already exist on disk. It prints an error message if a name exists on both disk and backup media, and continues.

-r

Specifies replace mode. **rbak** deletes the existing disk object, and replaces it with the object read from backup media.

-force

Forces object deletion if you have owner rights.

-du

Deletes when unlocked. If the object to be deleted is locked when **rbak** is invoked, the delete operation is performed when the object is unlocked.

-ms

Specifies merge-source mode. Similar to replace mode. If an object already exists on disk, **rbak** deletes the disk version and restores the backup media version (the source). However, if the object is a directory, **rbak** merges the back up media directory's contents with the disk directory.

-pr pn...

Preserves specified objects on the disk. Multiple pathnames and wildcarding are permitted. If the objects exist on disk, they are not overwritten by backup media versions. This option must be used with the **-ms** option.

-md

Specifies merge-destination mode. Similar to create mode. If an object already exists on disk (the destination) **rbak** does not restore the backup media version, and retains the disk version. However, if the object is a directory **rbak** merges the backup media directory's contents with the disk directory.

Label Control

-sla (default)

Displays the backup media file label on standard output.

-nsla

Does not display the backup media file label.

Listing Control

You may include the **-l** option, or any combination of **-ld**, **-lf**, and **-ll**.

- l** Writes all the file, directory, and link names to standard output.
- ld** Writes all directory names to standard output.
- lf** Writes all filenames to standard output.
- ll** Writes all linknames to standard output.

Backup Device Control

- anys** Forces **rbak** to accept any section of the backup file. When a backup file spans multiple backup media volumes, **rbak** normally begins with the backup media volume containing the backup file's first section, and proceeds to the backup media volume containing the second section, and so on. If you know which backup media volume contains the object you want to restore or index, use this option. This lets **rbak** start at any section of the backup file.
- reo** Forces previous volume to be reopened, and suppress reading of backup media volume label. Use only when backup media has not been repositioned since the last **wbak** or **rbak**.
- dev d[unit]** Specifies device type and unit number. *d* must be either **m** (for reel-to-reel magnetic tape, **ct** (for cartridge tape), or **f** (for floppy), depending on which drive is being used. *unit* is an integer (0-3). Both are required for reel-to-reel tapes (that is, **-dev m2**). A unit number is not required for floppy disks and cartridge tapes (that is, **-dev f**). If this option is omitted, **rbak** assumes device **m0**.

Note:

Floppy disk support for this command is limited. In particular, error detection during reads and writes is poor. Do not use this command with floppy disks when the data being placed on the floppy disks is critical and unrecoverable.

- from filename** Reads the backup input from a file written by **wbak** using the **-to** option. *filename* specifies the pathname of the file.
- stdin** Specifies the backup input media to be standard input. Used along with I/O redirection, this option is useful for reading files from foreign file systems.

- reten** Retensions the cartridge tape (unwind to the end, then rewind). This can be helpful if you encounter cartridge tape reading errors. Retensioning requires about 1.5 minutes to complete.
- nreten** (default) Do not retension the cartridge tape.
- unload** Unload the tape at the end of the backup session. This option is valid for reel-to-reel magnetic tapes only.
- rewind** Rewind the tape after reading or indexing. If this option is omitted, the tape is left positioned to the next tape file.

ACL Control

- dacl** (default) Assigns the destination directory's default ACL to the object being restored.
- sacl** Retains the restored object's original ACL.
- pdt** Preserves the object's original date-time modified and date-time used.

EXAMPLES

```
$ rbak -f 1 fred/soup
```

Read **fred/soup** in backup file 1 and restore it to disk. **fred/soup** may be a directory, file, or link.

```
$ rbak -f 1 fred/soup -as //node5/noodle
```

Restore **fred/soup** and place it in **noodle** on node5.

```
$ rbak -dev ct -rewind
```

Rewind the cartridge tape prior to removing it from the tape unit.

```
$ rbak src -from /fred/bck_out.file
```

Restore the directory **src** to disk. Read the backup input from the file **/fred/bck_out.file**, that should be written by **wbak** using the **-stdout** or **-from** option.

SEE ALSO

wbak(1), rwmt(1)

NAME

rcp – remote file copy

SYNOPSIS

```
rcp [ -p ] file1 file2
rcp [ -p ] [ -r ] file ... directory
```

DESCRIPTION

The **rcp** command copies files between machines. Each *file* or *directory* argument is either a remote file name of the form *rhost:path*, or a local filename containing no colon characters (:) or a slash mark (/) before any colons (:).

By default, **rcp** preserves the mode and owner of *file2* if it already existed; otherwise the mode of the source file modified by the **umask**(2) on the destination host is used.

If *path* is not a full path name, it is interpreted relative to your login directory on *rhost*. A *path* on a remote host may be quoted by using a backslash (\), double quotes ("), or a single quote ('), so that the metacharacters are interpreted remotely.

rcp does not prompt for passwords; your current local user name must exist on *rhost* and allow remote command execution via **rsh**(1C).

rcp handles third party copies, where neither source nor target files are on the current machine. Hostnames may also take the form *rname@rhost* to use *rname* rather than the current user name on the remote host. The destination hostname may also take the form *host.rname* to support destination machines that are running 4.2BSD versions of **rcp**.

OPTIONS

- r** If any of the source files are directories, copy each subtree rooted at that name; in this case the destination must be a directory.
- p** Attempt to preserve (duplicate) in copies the modification times and modes of the source files, ignoring the **umask**.

BUGS

Doesn't detect all cases where the target of a copy might be a file in cases where only a directory should be legal.

Is confused by any output generated by commands in a .login, .profile, or .cshrc file on the remote host.

SEE ALSO

cp(1), **ftp**(1C), **rsh**(1C), **rlogin**(1C)

NAME

rdist – remote file distribution program

SYNOPSIS

```
rdist [ -nqbRhivwy ] [ -f distfile ] [ -d var=value ] [ -m host ] [ name ... ]
rdist [ -nqbRhivwy ] -c name ... [login@]host[:dest]
```

DESCRIPTION

rdist is a program to maintain identical copies of files over multiple hosts. It preserves the owner, group, mode, and mtime of files if possible and can update programs that are executing. **rdist** reads commands from *distfile* to direct the updating of files and/or directories. If *distfile* is **-**, the standard input is used. If no **-f** option is present, the program looks first for **distfile**, then **Distfile** to use as the input. If no names are specified on the command line, **rdist** updates all of the files and directories listed in *distfile*. Otherwise, the argument is taken to be the name of a file to be updated or the label of a command to execute. If label and filenames conflict, it is assumed to be a label. These may be used together to update specific files using specific commands.

The **-c** option forces **rdist** to interpret the remaining arguments as a small *distfile*. The equivalent *distfile* is as follows.

```
( name ... ) -> [login@]host
      install [dest];
```

OPTIONS

Other options:

- n** Print the commands without executing them. This option is useful for debugging *distfile*.
- q** Quiet mode. Files that are being modified are normally printed on standard output. The **-q** option suppresses this.
- b** Binary comparison. Perform a binary comparison and update files if they differ rather than comparing dates and sizes.
- R** Remove extraneous files. If a directory is being updated, any files that exist on the remote host that do not exist in the master directory are removed. This is useful for maintaining truly identical copies of directories.
- h** Follow symbolic links. Copy the file that the link points to rather than the link itself.
- i** Ignore unresolved links. **rdist** will normally try to maintain the link structure of files being transferred and warn the user if all the links cannot be found.
- v** Verify that the files are up to date on all the hosts. Any files that are out of date will be displayed but no files will be changed nor any mail sent.

- w** Whole mode. The whole filename is appended to the destination directory name. Normally, only the last component of a name is used when renaming files. This will preserve the directory structure of the files being copied instead of flattening the directory structure. For example, renaming a list of files such as (**dir1/f1** **dir2/f2**) to **dir3** would create files **dir3/dir1/f1** and **dir3/dir2/f2**, instead of **dir3/f1** and **dir3/f1**.
- y** Younger mode. Files are normally updated if their *mtime* and *size* (see **stat(2)**) disagree. The **-y** option causes **rdist** not to update files that are younger than the master copy. This can be used to prevent newer copies on other hosts from being replaced. A warning message is printed for files which are newer than the master copy.
- d var=value** Define *var* to have *value*. The **-d** option is used to define or override variable definitions in the *distfile*. *Value* can be the empty string, one name, or a list of names surrounded by parentheses and separated by tabs and/or spaces.
- m host** Limit which machines are to be updated. Multiple **-m** arguments can be given to limit updates to a subset of the hosts listed the *distfile*.

Distfile contains a sequence of entries that specify the files to be copied, the destination hosts, and what operations to perform to do the updating. Each entry has one of the following formats.

```
<variable name> = <name list>
[ label: ] <source list> -> <destination list> <command list>
[ label: ] <source list> :: <time_stamp file> <command list>
```

The first format is used for defining variables. The second format is used for distributing files to other hosts. The third format is used for making lists of files that have been changed since some given date. The *source list* specifies a list of files and/or directories on the local host which are to be used as the master copy for distribution. The *destination list* is the list of hosts to which these files are to be copied. Each file in the source list is added to a list of changes if the file is out of date on the host which is being updated (second format) or the file is newer than the time stamp file (third format).

Labels are optional. They are used to identify a command for partial updates.

Newlines, tabs, and blanks are only used as separators and are otherwise ignored. Comments begin with **#** and end with a newline.

Variables to be expanded begin with **\$** followed by one character or a name enclosed in curly braces (see the examples at the end).

The source and destination lists have the following format:

<name>

or

(<zero or more names separated by white-space>)

The shell metacharacters [,] , { , } , * , and ? are recognized and expanded (on the local host only) in the same way as `csh(1)`. They can be escaped with a backslash. The ``` character is also expanded in the same way as `csh` but is expanded separately on the local and destination hosts. When the `-w` option is used with a filename that begins with ```, everything except the home directory is appended to the destination name. File names which do not begin with `/` or ``` use the destination user's home directory as the root directory for the rest of the filename.

The command list consists of zero or more commands of the following format.

```
install <options> opt_dest_name ;
notify  <name list> ;
except  <name list> ;
except_pat      <pattern list> ;
special <name list>string ;
```

The **install** command is used to copy out of date files and/or directories. Each source file is copied to each host in the destination list. Directories are recursively copied in the same way. *opt_dest_name* is an optional parameter to rename files. If no **install** command appears in the command list or the destination name is not specified, the source filename is used. Directories in the pathname will be created if they do not exist on the remote host. To help prevent disasters, a non-empty directory on a target host will never be replaced with a regular file or a symbolic link. However, under the `-R` option a non-empty directory will be removed if the corresponding filename is completely absent on the master host. The *options* are `-R`, `-h`, `-i`, `-v`, `-w`, `-y`, and `-b` and have the same semantics as options on the command line except they only apply to the files in the source list. The login name used on the destination host is the same as the local host unless the destination name is of the format *login@host*.

The **notify** command is used to mail the list of files updated (and any errors that may have occurred) to the listed names. If no `@` appears in the name, the destination host is appended to the name (e.g., *name1@host*, *name2@host*, ...).

The **except** command is used to update all of the files in the source list except for the files listed in *name list*. This is usually used to copy everything in a directory except certain files.

The **except_pat** command is like the **except** command except that *pattern list* is a list of regular expressions (see `ed(1)` for details). If one of the patterns matches some string within a filename, that file will be ignored. Note that since `\` is a quote character, it must be doubled to become part of the regular expression. Variables are expanded in *pattern list* but not shell file pattern matching characters. To include a `$`, it must be escaped with `\`.

The **special** command is used to specify `sh(1)` commands that are to be executed on the remote host after the file in *name list* is updated or installed. If the *name list* is

omitted then the shell commands will be executed for every file updated or installed. The shell variable **FILE** is set to the current filename before executing the commands in *string*. *string* starts and ends with " and can cross multiple lines in *distfile*. Multiple commands to the shell should be separated by the ; character. Commands are executed in the user's home directory on the host being updated. The **special** command can be used to rebuild private databases, etc. after a program has been updated.

EXAMPLE

```
HOSTS = ( matisse root@arpa)

FILES = ( /bin /lib /usr/bin /usr/games
          /usr/include/{*.h,{stand,sys,vax*,pascal,machine}/*.h}
          /usr/lib /usr/man/man? /usr/ucb /usr/local/rdist )

EXLIB = ( Mail.rc aliases aliases.dir aliases.pag crontab dshrc
          sendmail.cf sendmail.fc sendmail.hf sendmail.st uucp vfont )

${FILES} -> ${HOSTS}
    install -R ;
    except /usr/lib/${EXLIB} ;
    except /usr/games/lib ;
    special /usr/lib/sendmail "/usr/lib/sendmail -bz" ;

srcs:
/usr/src/bin -> arpa
    except_pat ( \\o.$ /SCCS\$ ) ;

IMAGEN = (ips dviimp catdvi)

imagen:
/usr/local/${IMAGEN} -> arpa
    install /usr/local/lib ;
    notify ralph ;

${FILES} :: stamp.cory
    notify root@cory ;
```

FILES

distfile input command file
/tmp/rdist* temporary file for update lists

DIAGNOSTICS

A complaint about mismatch of **rdist** version numbers may really stem from some problem with starting your shell, e.g., you are in too many groups.

BUGS

Source files must reside on the local host where **rdist** is executed.

There is no easy way to have a special command executed after all files in a directory have been updated.

Variable expansion only works for name lists; there should be a general macro facility.

rdist aborts on files which have a negative mtime (before Jan 1, 1970).

There should be a “force” option to allow replacement of non-empty directories by regular files or symlinks. A means of updating file modes and owners of otherwise identical files is also needed.

SEE ALSO

sh(1), **csh(1)**, **rcp(1)**, **stat(2)**

NAME

refer – find and insert literature references in documents

SYNOPSIS

refer [*options*] [*file* ...]

DESCRIPTION

refer is a preprocessor for **nroff**(1) or **troff**(1) that finds and formats references for footnotes or endnotes. It is also the base for a series of programs designed to index, search, sort, and print standalone bibliographies, or other data entered in the appropriate form.

Given an incomplete citation with sufficiently precise keywords, **refer** will search a bibliographic database for references containing these keywords anywhere in the title, author, journal, etc. The input file (or standard input) is copied to standard output, except for lines between `.[` and `.]` delimiters. These delimiters are assumed to contain keywords, and are replaced by information from the bibliographic database. You may also search different databases, override particular fields, or add new fields. The reference data, from whatever source, are assigned to a set of **troff** strings. Macro packages such as **ms**(7) print the finished reference text from these strings. By default, references are flagged by footnote numbers.

References can be searched more rapidly by running **indxbib**(1) on them before using **refer**; failure to index results in a linear search. When **refer** is used with the **eqn**, **neqn** or **tbl** preprocessors, **refer** should be first, to minimize the volume of data passed through pipes.

To use your own references, put them in the format described below. The **refer** preprocessor and associated programs expect input from a file of references composed of records separated by blank lines. A record is a set of lines (fields), each containing one kind of information. Fields start on a line beginning with a percent sign (%), followed by a key-letter, then a blank, and finally the contents of the field, and continue until the next line starting with a percent sign. The output ordering and formatting of fields is controlled by the macros specified for **nroff/troff** (for footnotes and endnotes) or **roffbib** (for stand-alone bibliographies). For a list of the most common key-letters and their corresponding fields, see **addbib**(1).

OPTIONS

The following options are available:

- an** Reverse the first *n* author names (Jones, J. A. instead of J. A. Jones). If *n* is omitted, all author names are reversed.
- b** Bare mode: do not put any flags in text (neither numbers nor labels).
- ckey** Capitalize (with CAPS SMALL CAPS) the fields whose key-letters are in *keys*.

- e Instead of leaving the references where encountered, accumulate them until a sequence of the form
 .
 \$LIST\$
 .]
 is encountered, and then write out all references collected so far. Collapse references to same source.
- fn Set the footnote number to *n* instead of the default of 1 (one). With labels rather than numbers, this flag is a no-op.
- kx Instead of numbering references, use labels as specified in a reference data line beginning %x; by default x is L.
- lm,n Instead of numbering references, use labels made from the senior author's last name and the year of publication. Only the first *m* letters of the last name and the last *n* digits of the date are used. If either *m* or *n* is omitted, the entire name or date, respectively, is used.
- n Do not search the default file /usr/dict/papers/Ind. If there is a REFER environment variable, the specified file will be searched instead of the default file; in this case the -n flag has no effect.
- p *bib* Take the next argument *bib* as a file of references to be searched. The default file is searched last.
- skeys Sort references by fields whose key-letters are in the *keys* string; permute reference numbers in text accordingly. Implies -e. The key-letters in *keys* may be followed by a number to indicate how many such fields are used, with + taken as a very large number. The default is AD which sorts on the senior author and then date; to sort, for example, on all authors and then title, use -sA+T.
- Bl.m Bibliography mode. Take a file composed of records separated by blank lines, and turn them into troff input. Label *l* will be turned into the macro .m with *l* defaulting to %X and .m defaulting to .AP (annotation paragraph).
- P Place punctuation marks .,:;?! after the reference signal, rather than before. (Periods and commas used to be done with strings.)
- S Produce references in the Natural or Social Science format.

EXAMPLE

The following example shows the format of a **refer** entry.

```
%A    M. E. Lesk
%T    Some Applications of Inverted Indexes on the UNIX System
%B    UNIX Programmer's Manual
%V    2b
%I    Bell Laboratories
%C    Murray Hill, NJ
%D    1978
```

FILES

```
/usr/dict/papers    directory of default publication lists
/usr/lib/refer       directory of companion programs
```

BUGS

Blank spaces at the end of lines in bibliography fields will cause the records to sort and reverse incorrectly. Sorting large numbers of references causes a core dump.

SEE ALSO

addbib(1), sortbib(1), roffbib(1), indxbib(1), lookbib(1)

NAME

rev – reverse lines of a file

SYNOPSIS

rev [*file*] ...

DESCRIPTION

rev copies the named files to the standard output, reversing the order of characters in every line. If no file is specified, **rev** reads the standard input.

NAME

rlogin – remote login

SYNOPSIS

```
rlogin rhost [ -ec ] [ -8 ] [ -L ] [ -l username ]
rhost [ -ec ] [ -8 ] [ -L ] [ -l username ]
```

DESCRIPTION

rlogin connects your terminal on the current local host system *lhost* to the remote host system *rhost*. The second form of the command shown in the synopsis requires that you place host names as commands in the directory */usr/hosts*, and put this directory in your search path. You can then omit **rlogin** and enter only the host name (*rhost*) to remotely log in to a remote host.

Each host has a file, */etc/hosts.equiv*, that contains a list of *rhhosts* with which it shares account names. (The host names must be the standard names as described in *rsh(1C)*.) When you execute **rlogin** as the same user on an equivalent host, you don't need to provide a password. Each user may also have a private equivalence list in a file *.rhosts* in his or her log-in directory. Each line in this file should contain an *rhost* and a *username* separated by a space, giving additional cases where logins without passwords are to be permitted. Note however, if your password has been overridden on the remote host, you will be prompted for a password regardless of how you log in.

If the originating user is not equivalent to the remote user, then a login and password will be prompted for on the remote machine as in *login(1)*. To avoid some security problems, the *.rhosts* file must be owned by either the remote user or root.

The remote terminal type is the same as your local terminal type (as given in your environment *TERM* variable). The terminal or window size is also copied to the remote system if the server supports the option, and changes in size are reflected as well. All echoing takes place at the remote site, so that (except for delays) the **rlogin** is transparent. Flow control via *CTRL/S* and *CTRL/Q* and flushing of input and output on interrupts are handled properly.

A line of the form “~.” (a tilde followed by a period) disconnects from the remote host, where tilde is the escape character. Similarly, the line “~Z” (where *~Z*, *CTRL/Z*, is the suspend character) will suspend the **rlogin** session. Substitution of the delayed-suspend character (normally *^Y*) for the suspend character suspends the send portion of the **rlogin**, but allows output from the remote system. Use the **-e** option to specify a different escape character.

OPTIONS

- ec** Specify *c* as the escape character to use. There is no space separating **-e** and the argument character.
- 8** Allows an eight-bit input data path at all times; otherwise parity bits are stripped except when the remote side's stop and start characters are other than *CTRL/S* and *CTRL/Q*.
- L** Allows the **rlogin** session to be run in litout mode.

-l *username* Specify a different *username*. This is necessary when the originating user is not equivalent to the remote user.

FILES

/usr/hosts/* for **rhost** version of the command

BUGS

More of the environment should be propagated.

SEE ALSO

rsh(1C)

NAME

rm, rmdir – remove (unlink) files or directories

SYNOPSIS

rm [-frl] [-] *file* ...

rmdir *dir* ...

DESCRIPTION

The **rm** command 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 you do not have write permission on a file and the standard input is a terminal, **rm** prints the file's permissions and reads a line from the standard input. If that line begins with "y", the file is deleted; otherwise, the file remains. If a designated file is a directory, **rm** prints an error comment unless you use the **-r** option.

The **rmdir** command removes entries for the named directories, which must be empty.

OPTIONS

- | | |
|-----------|--|
| -f | Force. Suppress questions and don't report errors. |
| -r | Recursively delete the entire contents of the specified directory and the directory itself. |
| -l | Interactive; ask whether to delete each file. With the -r option, ask whether to examine each directory. |
| - | Indicate that all the arguments following it are to be treated as filenames. This option allows the specification of filenames starting with a minus sign (-). |

SEE ALSO

unlink(2), rmdir(2)

NAME

rmail – handle remote mail received via **uucp**

SYNOPSIS

rmail *user* ...

DESCRIPTION

rmail interprets incoming mail received via **uucp**(1C), collapsing “From” lines in the form generated by **binmail**(1) into a single line of the form *return-path!sender*, and passing the processed mail on to **sendmail**(8).

rmail is explicitly designed for use with **uucp** and **sendmail**.

BUGS

rmail should not reside in **/bin**.

SEE ALSO

binmail(1), **uucp**(1C), **sendmail**(8)

NAME

rmdel – remove a delta from an SCCS file

SYNOPSIS

rmdel -rSID [*file* ...] [*directory*]

DESCRIPTION

rmdel removes the delta (specified by the SCCS IDentification number, or SID) from each named SCCS *file*. The delta to be removed must be the newest (most recent) one in its branch in the delta chain. The SID specified must not be that of a version being edited for the purpose of making a delta. Thus, if a *p-file* exists for the named SCCS file, the SID specified must *not* appear in any entry of the *p-file*. Refer to **get(1)** for information about the creation of *p-files*.

If you give a directory as an argument, **rmdel** behaves as though each file in the directory were specified as a named file, except that it silently ignores any non-SCCS and unreadable files. If you supply a dash (-) in place of a filename, the standard input is read. Each line of the standard input is taken to be the name of an SCCS file to be processed; again, non-SCCS files and unreadable files are silently ignored.

Permission to remove a delta is granted to the person who created it, as well as to the owner of the associated file and directory.

FILES

<i>x.file</i>	see delta(1)
<i>z.file</i>	see delta(1)

DIAGNOSTICS

Use **help(1)** for explanations.

SEE ALSO

delta(1), **get(1)**, **help(1)**, **prs(1)**, **sccs(1)**

NAME

rmdir, rm - remove (unlink) directories or files

SYNOPSIS

rmdir *dir* ...

rm [**-f**] [**-r**] [**-i**] [**-**] *file* ...

DESCRIPTION

The **rmdir** command removes entries for the named directories, which must be empty.

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.

OPTIONS

- | | |
|-----------|---|
| -f | Suppress questions and don't report errors. |
| -r | Recursively delete the entire contents of the specified directory, and the directory itself. |
| -i | Interactive. Ask whether to delete each file, and, when used with -r , whether to examine each directory. |
| - | The null option; indicates that all the arguments following it are to be treated as filenames. This option allows the specification of filenames starting with a minus (-). |

SEE ALSO

rm(1), **unlink**(2), **rmdir**(2)

NAME

roffbib – run off bibliographic database

SYNOPSIS

```
roffbib [ -e ] [ -h ] [ -nN ] [ -o ] [ -r ] [ -s ] [ -Tterm ] [ -x ] [ -m mac ]
[ -V ] [ -Q ] [ file ... ]
```

DESCRIPTION

roffbib prints out all records in a bibliographic database, in bibliography format rather than as footnotes or endnotes. Generally it is used in conjunction with **sortbib**, as shown below:

```
sortbib database | roffbib
```

roffbib accepts most of the options that **nroff**(1), understands, most importantly the **-T** flag to specify terminal type.

If you enter abstracts or comments following the %X field key, **roffbib** formats them into paragraphs for an annotated bibliography. You may supply several %X fields if several annotation paragraphs are desired.

OPTIONS

- e** Produce equally-spaced words in adjusted lines, using full terminal resolution.
- h** Use output tabs during horizontal spacing to speed output and reduce the output character count. Assume tab setting to be every nominal character width.
- nN** Make *N* the page number of the first page generated.
- x** Suppress the printing of these abstracts.
- m mac** Specify a user-defined set of macros to replace the ones defined in */usr/lib/tmac/tmac.bib*. There should be a space between the **-m** and the macro filename.
- V** Send output to a Versatec printer.
- Q** Queue output for phototypesetter.

Five command-line registers control formatting style of the bibliography, much like the number registers of **ms**(7).

- rN1** Number the references starting at one (1).
- rV2** Double space the bibliography,
- rV1** Double space references but single space annotation paragraphs.
- rL6i** Change the line length from the default 6.5 inches to 6 inches.
- rO1i** Change the page offset from the default of 0 to one inch (Use a capital O, not zero). Note: with the **-V** and **-Q** flags the default page offset is already one inch.

BUGS

Users have to rewrite macros to create customized formats.

FILES

/usr/lib/tmac/tmac.bib file of macros used by **nroff/troff**

SEE ALSO

refer(1), **addbib(1)**, **sortbib(1)**, **indxbib(1)**, **lookbib(1)**, **nroff(1)**, **troff(1)**

NAME

rsh – remote shell

SYNOPSIS

```
rsh host [ -l username ] [ -n ] command
host [ -l username ] [ -n ] command
```

DESCRIPTION

rsh connects to the specified *host*, and executes the specified *command*. **rsh** copies its standard input to the remote command, the standard output of the remote command to its standard output, and the standard error of the remote command to its standard error. Interrupt, quit and terminate signals are propagated to the remote command; **rsh** normally terminates when the remote command does.

The remote username used is the same as your local username, unless you specify a different remote name with the **-l** option.

If you omit *command*, then instead of executing a single command, you will be logged in on the remote host using **rlogin(1C)**.

Shell metacharacters which are not quoted are interpreted on the local machine, while quoted metacharacters are interpreted on the remote machine.

Host names are given in the file */etc/hosts*. Each host has one standard name (the first name given in the file), which is rather long and unambiguous, and optionally one or more nicknames. The host names for local machines are also commands in the directory */usr/hosts*; if you put this directory in your search path, then the **rsh** can be omitted.

OPTIONS

- l** *username* Specify a remote *username* different from your local username. This remote name must be equivalent (in the sense of **rlogin(1C)**) to the originating account; no provision is made for specifying a password with a command.
- n** Redirect the input of **rsh** to */dev/null*.

EXAMPLES

The following command appends the remote file **remotefile** to the localfile **localfile**.

```
rsh otherhost cat remotefile >> localfile
```

The command below appends **remotefile** to **otherremotefile**.

```
rsh otherhost cat remotefile ">>" otherremotefile
```

BUGS

If you are using **cs(1)** and put a **rsh(1C)** in the background without redirecting its input away from the terminal, it will block even if no reads are posted by the remote command. If no input is desired you should redirect the input of **rsh** to */dev/null* using the **-n** option.

You cannot run an interactive command (such as **rogue(6)** or **vi(1)**); use **rlogin(1C)**.

Stop signals stop the local **rsh** process only; this is arguably wrong, but currently hard to fix for reasons too complicated to explain here.

RSH(1C)

Domain/OS BSD

RSH(1C)

FILES

/etc/hosts

/usr/hosts/*

SEE ALSO

rlogin(1C)

NAME

runtime – show host status of local machines

SYNOPSIS

runtime [**-a**] [**-r**] [**-t**] [**-u**]

DESCRIPTION

The **runtime** command gives a status line like **uptime(1)** for each machine on the local network; these are formed from packets broadcast by each host on the network once a minute.

Machines for which no status report has been received for 11 minutes are shown as being down. Users idle an hour or more are not counted unless the **-a** flag is given. Normally, the listing is sorted by host name.

OPTIONS

- a** Count the users who've been idle for an hour or more.
- r** Reverse the sort order.
- t** Sort the listing by uptime.
- u** Sort the listing by number of users.

FILES

/usr/spool/rwho/whod.* data files

SEE ALSO

rwho(1C)

NAME

rwho – who's logged in on local machines

SYNOPSIS

rwho [-a]

DESCRIPTION

The **rwho** command produces output similar to **who**, but for all machines on the local network. If no report has been received from a machine for five minutes, **rwho** assumes the machine is down, and does not report users last known to be logged into that machine.

If a user hasn't typed to the system for a minute or more, **rwho** reports this idle time. If a user hasn't typed to the system for an hour or more, the user is omitted from the output of **rwho**, unless you specified the **-a** flag.

BUGS

The **rwho** command is unwieldy when the number of machines on the local net is large.

FILES

/usr/spool/rwho/whod.* information about other machines

SEE ALSO

runtime(1C), **rwhod(8C)**,
Managing BSD System Software.

NAME

rwmt – read/write foreign magtapes

SYNOPSIS

rwmt [*option*]... [-p] [-r|-w|-l|-l] [*pathname*]...

DESCRIPTION

rwmt reads tapes from non-Domain installations and writes tapes that can be read by non-Domain installations. **rwmt** can read and write unlabeled tapes, as well as ANSI level 1–4 labeled tapes.

pathname (optional)

Specify the name of file to be read from or written to tape. This argument is valid only with the **-r** and **-w** mode-control options (below). Multiple pathnames are permitted. Wildcarding is permitted for write (**-w**) operations only.

Default if omitted: read pathnames from standard input

OPTIONS

Mode control

You must specify one of the following mode-control options. If you omit this option, **rwmt** prompts you for it. The **-p** option tells **rwmt** to prompt for all necessary options.

-l[abel]

Write ANSI X3.27–1978 volume label on a tape. This option causes **rwmt** to write an ANSI volume label and dummy file on the magtape volume. You may specify an optional owner and volume ID, which are stored in the volume label. (see **-vid** and **-own** below. This is the way to initialize a labeled tape; if any information existed on the tape, it is erased by this labeling operation.

If you are labeling a tape, you can also use the following two options.

-vid vol_id

Specifies a 1–6 character volume ID for use when labeling a volume. This option is valid only when used with the **-l** mode-control option (above). The default volume ID is ' ' (blank).

-own owner_id

Specifies a 1–14 character owner ID for use when labeling a volume. This option is valid only when used with the **-l** mode-control option (above). The default owner ID is ' ' (blank).

- i[ndex]** Lists objects on an ANSI-labeled physical tape volume. **-index** produces a listing of all files or file sections on an ANSI-labeled physical tape volume. The contents of the physical volume (VOL1) label and all file header labels are written to standard output.
- w[rite]** Specifies one or more disk files (*pathname* argument) to be written to tape. The default format is ANSI labeled, ASCII, fixed-length records of 80 bytes each, and 80-byte blocks. If desired, any of these parameters can be changed using the options described below. If more than one *pathname* is specified, the disk files are written to sequential tape files. Tapes written by **rwmt** are always in accordance with ANSI level 4 format. Before writing a labeled file, the tape volume itself must be labeled with the **-label** mode-control option (above).
- r[ead]** Specifies one or more tape files to be read from tape and stored on disk. **read** reads one or more tape files and writes them to disk using the specified *pathnames* (*pathname* argument). The default tape file format is the same as that for the **write** option. If the tape is labeled under ANSI level 2, 3, or 4, the file format (block length, record length, and record format) is read from the tape. If the tape is unlabeled, or labeled with ANSI level 1, you must specify the tape format using the options below. If more than one *pathname* is specified, adjacent tape files are read and stored under the specified *pathnames*.

Label Control

- ansi** (default) Specifies that the tape is labeled in conformance to ANSI X3.27-1978, level 1, 2, 3, or 4.
- unlab** Specifies that the tape is unlabeled. Data spanning physical volumes is not supported on unlabeled tapes.

Tape Format

- asc** (default) Specifies that all tape file contents are in ASCII characters.
- ebc** Specifies that all tape file contents (except labels) are in EBCDIC characters.
- raw** Specifies that all tape file data is to be treated in raw form.
- npar** (default) Specifies no disturbance of parity bits when reading or writing data.
- par** Specifies that parity bits should be forced off when reading data from tape and forced on when writing data to tape.
- rl *reclen*** Specifies the maximum length, in bytes, of a record in the tape file. This option is valid only when used with either the **-r** or the **-w** mode-control options (above). It is unnecessary when reading an ANSI level 2, 3, or 4 file. The default record length is 80 bytes.

- bl *blocklen*** Specifies the length, in bytes, of a physical tape block. This option is valid only when used with either the **-r** or the **-w** mode-control options (above). It is unnecessary when reading an ANSI level 2, 3, or 4 file. The default block length is 80 bytes.
- bf *blockfac*** Specifies a blocking factor — the number of records to store into or read from a physical tape block. This is an alternative to the **-bl** option, since the record length multiplied by the blocking factor yields the block length. This option is valid only when used with either the **-r** or **-w** mode-control options (above). Using this option is meaningful only if your tape has fixed-length records. This option is unnecessary when reading an ANSI level 2, 3, or 4 file. The default blocking factor is 1.
- rf *format*** Specifies record format. Valid values for *format* are **f** (fixed-length records and blocks); **d** (variable-length records (this is ANSI 'D' format)); **s** (spanned records); or **u** (undefined record format). The default format is **f**. Note that if you are writing a cartridge tape, only 512 byte blocks may be written; **d**, **s**, and **u** formats are not supported.

Tape File Identifiers

- fid *file_id*** Specifies a 1–17 character file ID to be written in the file header label for use when writing a file to a labeled volume. This option is valid only when used with the **-w** mode-control option (above). If this option is omitted, the name of the file being written is used.
- f [*position*]** Specifies the file position for **-r** or **-w** operations. Valid values for *position* are **cur**, **end**, or a nonzero integer value. A position of **cur** selects the current tape position; the tape must have been previously read or written by **rwmt** and its position must not have been disturbed. This option is valid only when used with either the **-r** or the **-w** mode-control options (above).

A position of **end** selects the end of the tape file set. This option is valid only when used with the **-w** mode-control option, and causes **rwmt** to append the specified disk file (*pathname* argument) to the very end of the tape file set.

A position specified by a nonzero integer value selects the file at that absolute position in the tape volume. This option is valid only when used with either the **-r** or **-w** mode-control options (above). If multiple *pathname* arguments are supplied, the value of *position* is incremented by one after each file has been read or written.

The default value for *position* is 1.

Backup Device Control

- dev *d[unit]*** Specifies device type and unit number. *d* must be either **m** (for reel-to-reel magnetic tape), **ct** (for cartridge tape), or **f** (for floppy), depending on which drive is being used. *unit* is an integer (0-3). Both are required for reel-to-reel tapes (that is, **-dev m2**). A unit number is not required for floppy disks and cartridge tapes (that is, **-dev f**). If this option is omitted, **rbak** assumes device **m0**.
- nobs** Specifies that byte swapping should not be done in software. This operation is valid for magnetic tapes only. On the multibus data gets byte swapped. **rwmt** does byte swapping in software so that the tape gets written out in the correct machine order. **wbak** and **rbak** do not do byte swapping in software, as a result the two swaps done by the multibus cancel out. This option is useful in writing to a magnetic tape an intermediate file to which **wbak** output has been directed. Byte swapping should not be done by **rwmt** if the intermediate file written by **wbak** is now written raw to the magnetic tape using **rwmt**.
- reten** Retensions the cartridge tape (unwind to the end, then rewind). This can be helpful if you have encountered cartridge-tape reading errors. Retensioning requires about 1.5 minutes to complete.
- nreten** (default) Does not retension the cartridge tape.

Miscellaneous Control Options

- sbin** Causes all stream files written to contain the binary attribute (normally, output stream files contain the ASCII attribute).
- p** Causes **rwmt** to prompt for all unspecified parameters.

EXAMPLES

Initialize a tape with the given owner and volume ID.

```
$ rwmt -label -own "R AND D" -vid "demo"
```

List the files on the tape.

```
$ rwmt -index
```

Volume label:

```
Volume ID: "DEMO"           Owner ID: "R AND D"           Access: " "
File  Section  File ID      Cr Date      Acc RF  RL  BL
  1      1      CMF_EXAMPLE  83/02/17      D   D   200 2048
  2      1      CMT_EXAMPLE  83/02/17      D   D   200 2048
  3      1      CPBOOT_EXAMPLE 83/02/17      D   D   200 2048
  4      1      CPF_EXAMPLE  83/02/17      D   D   200 2048
  5      1      CPT_EXAMPLE  83/02/17      D   D   200 2048
End of file set.
$
```

Copy tape file 3 to a disk file named **cpboot_example.tape**.

```
$ rwmt -r cpboot_example.tape -f 3
4 records read from tape file 3 into
  "cpboot_example.tape".
$
```

rwmt permits a tape file to be read in "raw" mode. In this mode, each block read from the tape is written into one record in a stream file, unmodified by the program. To read a file in "raw" mode, you should specify the maximum record size using the **-rl** argument. If you do not, the default value of 80 bytes is used, and any records longer than that are truncated. Also, undefined record format should be used. For example

```
$ rwmt -r -f 1 -rf u -raw -rl 512 rawfile
```

reads tape file number 1 into **rawfile**, with a maximum record length of 512 bytes.

Files may be written in the same manner:

```
$ rwmt -w -f 1 -rf u -raw -rl 512 rawfile
```

The file **//backup/tmp1** is written out to the magnetic tape in "raw" mode. The record length is specified to be 8k and no byte swapping is done in software. This is useful for writing out an intermediate file to which **wbak** has written its output. Note that all tapes written by **rwmt** must have a ANSI standard volume label for **rbak** to be able to read the tape

```
rwmt -w -f 1 -raw -rl 8192 -nobs //backup/tmp1 -ansi
```

If **rwmt** writes a file with **-nobs** option, you should use **-nobs** option to read it using **rwmt**.

SEE ALSO

rbak(1), **wbak(1)**

NAME

sact – print current SCCS file editing activity

SYNOPSIS

sact *file* ...

DESCRIPTION

sact informs you of any impending deltas to a named SCCS *file*. This information is most helpful if you mistakenly executed a **get**(1) command using the **-e** option without a subsequent execution of **delta**(1). If a directory is named on the command line, **sact** behaves as though each file in the directory were specified as a named file, except that non-SCCS files and unreadable files are silently ignored. If you specify a dash (–) instead of a filename, **sact** reads the standard input, interpreting each line as the name of an SCCS file to be processed.

The output for each named file consists of five fields separated by spaces. These fields specify the following information:

Field 1	SID of an existing delta in the SCCS file to which changes will be made to create the new delta.
Field 2	SID for the new delta to be created.
Field 3	Log name of the user who will make the delta; i.e., person who executed a get (1) for editing.
Field 4	Date that the get -e command was executed.
Field 5	Time that the get -e command was executed.

DIAGNOSTICS

Use **help**(1) for explanations.

SEE ALSO

delta(1), **get**(1), **sccs**(1), **unget**(1)

NAME

sccs – front end for the SCCS subsystem

SYNOPSIS

sccs [**-r**] [**-dpath**] [**-ppath**] *command* [*flags*] [*arg* ...]

DESCRIPTION

The **sccs** command is a front end to the Source Code Control System (SCCS) programs that helps them mesh more cleanly with the rest of UNIX. It also includes the capability to run "set user ID" to another user to provide additional protection.

Basically, **sccs** runs the *command* with the specified *flags* and *args*. Each argument is normally modified to be prefixed with **SCCS/s**.

Flags to be interpreted by the **sccs** program must appear before the *command* argument. Flags to be passed to the actual SCCS program must come after the *command* argument. These flags are specific to the command and are discussed in the documentation for that command. The SEE ALSO section lists the standard SCCS commands that are documented in section one of this manual. For more information about the standard SCCS commands, see the documentation for that command.

Besides the usual SCCS commands, several pseudo commands can be issued. These pseudo commands are described in the following list.

edit	Equivalent to get -e
delget	Perform a delta on the named files and then get new versions. The new versions will have ID keywords expanded, and will not be editable. The -m , -p , -r , -s , and -y flags will be passed to delta, and the -b , -c , -e , -i , -k , -l , -s , and -x flags will be passed to get.
deledit	Equivalent to delget except that the get phase includes the -e flag. This option is useful for making a checkpoint of your current editing phase. The same flags will be passed to delta as described above, and all the flags listed for get above, except -e and -k , are passed to edit .
create	Creates an SCCS file, taking the initial contents from the file of the same name. Any flags to admin are accepted. If the creation is successful, the files are renamed with a comma on the front. These should be removed when you are convinced that the SCCS files have been created successfully.

- fix** Must be followed by a **-r** flag. This command essentially removes the named delta, but leaves you with a copy of the delta with the changes that were in it. It is useful for fixing small compiler bugs, etc. Since it doesn't leave audit trails, it should be used carefully.
- clean** This routine removes everything from the current directory that can be recreated from SCCS files. It will not remove any files being edited. If the **-b** flag is given, branches are ignored in the determination of whether they are being edited; this is dangerous if you are keeping the branches in the same directory.
- unedit** This is the opposite of an **edit** or a **get -e**. It should be used with extreme caution, since any changes you made since the **get** will be irretrievably lost.
- info** Gives a listing of all files being edited. If the **-b** flag is given, branches (i.e., SID's with two or fewer components) are ignored. If the **-u** flag is given (with an optional argument) then only files being edited by you (or the named user) are listed.
- check** Like **info** except that nothing is printed if nothing is being edited, and a non-zero exit status is returned if anything is being edited. The intent is to have this included in an "install" entry in a makefile to insure that everything is included into the SCCS file before a version is installed.
- tell** Gives a newline-separated list of the files being edited on the standard output. Takes the **-b** and **-u** flags like **info** and **check**.
- diffs** Gives a diff listing between the current version of the program(s) you have out for editing and the versions in SCCS format. The **-r**, **-c**, **-i**, **-x**, and **-t** flags are passed to **get**; the **-l**, **-s**, **-e**, **-f**, **-h**, and **-b** options are passed to **diff**. The **-C** flag is passed to **diff** as **-c**.
- print** This command prints out verbose information about the named files.

OPTIONS

- r** Runs **sccs** as the real user rather than as whatever effective user **sccs** is "set user ID" to.
- d** Gives a root directory for the SCCS files. The default is the current directory.
- p** Defines the pathname of the directory in which the SCCS files will be found; **SCCS** is the default.

The **-p** flag differs from the **-d** flag in that the **-d** argument is prepended to the entire pathname and the **-p** argument is inserted before the final component of the pathname.

For example:

```
sccs -d/x -py get a/b
```

will convert to:

```
get /x/a/y/s.b
```

The intent here is to create aliases such as:

```
alias syssecs sccs -d/usr/src
```

which will be used as:

```
syssecs get cmd/who.c
```

If the environment variable **PROJECT** is set, its value is used to determine the **-d flag**. If it begins with a slash, it is taken directly; otherwise, the home directory of a user of that name is examined for a subdirectory **src** or **source**. If such a directory is found, it is used.

Certain commands (such as **admin**) cannot be run "set user ID" by all users, since this would allow anyone to change the authorizations. These commands are always run as the real user.

EXAMPLES

To get a file for editing, edit it, and produce a new delta:

```
sccs get -e file.c
ex file.c
sccs delta file.c
```

To get a file from another directory:

```
sccs -p/usr/src/scs/s. get cc.c
```

or

```
sccs get /usr/src/scs/s.cc.c
```

To make a delta of a large number of files in the current directory:

```
sccs delta *.c
```

To get a list of files being edited that are not on branches:

```
sccs info -b
```

To delta everything being edited by you:

```
sccs delta `scs tell -u`
```

In a makefile, to get source files from an SCCS file if it does not already exist:

SRCS = <list of source files>

\$(SRCS):

sccs get \$(REL) \$@

NOTES

It should be able to take directory arguments on pseudo commands like the SCCS commands do.

SEE ALSO

admin(1), comb(1), delta(1), get(1), help(1), rmdel(1), sact(1), sccsdiff(1), what(1);
Domain/OS Programming Environment Reference

NAME

sccsdiff – compare two versions of an SCCS file

SYNOPSIS

sccsdiff *-rSID1 -rSID2* [*-p*] [*-sn*] *file* ...

DESCRIPTION

sccsdiff compares two versions of an SCCS file and generates the differences between the two versions. You may specify any number of SCCS files, but arguments will apply to all files.

OPTIONS

- rSID?* *SID1* and *SID2* specify the deltas of an SCCS file to be compared. Versions are passed to **bdiff**(1) in the order given.
- p* Pipe output for each file through **pr**(1).
- sn* Specify *n* as the file segment size that **bdiff** will pass to **diff**(1). This is useful when **diff**(1) fails due to a high system load.

FILES

/tmp/get???? Temporary files

DIAGNOSTICS

file : No differences The two versions are the same.

Use **help**(1) for explanations.

SEE ALSO

bdiff(1), **get**(1), **help**(1), **pr**(1) **sccs**(1)

NAME

scrattr – screen attributes

SYNOPSIS

scrattr [**-avcpxy**]

DESCRIPTION

Without any options, **scrattr** lists the X and Y dimensions of the display in pixels. Screen attributes are always listed in the same order: X coordinates, Y coordinates, number of planes, number of primary colors. This order is independent of the order in which the options are specified.

OPTIONS

- a** Display all attributes.
- v** Print a verbose description of each field, with the attributes on separate lines. (Without the **-v** option, attributes appear on the same line separated by tabs.) If any options other than **-v** are specified, only that combination of attributes will be displayed, and always in the canonical order given above.
- c** Display the number of primary colors on the display.
- p** Display the number of physical bit planes on the display.
- x** Display the X dimension of the display in pixels.
- y** Display the Y dimension of the display in pixels.

SEE ALSO

stty(1), tabs(1), tset(1), tty(1)

NAME

script – make typescript of terminal session

SYNOPSIS

script [**-a**] [*file*]

DESCRIPTION

The **script** command forks a new shell and makes a typescript of everything printed on your terminal (or a pad on a node). The typescript is written to (or appended to) *file*; if no *file* is specified, the typescript is saved in the file **typescript**. It can be sent to the line printer later with **lpr**. The **script** ends when the forked shell exits.

This program is useful when you're using a crt and wish to have a hard-copy record of the dialog.

OPTIONS

-a	Append to <i>file</i> . If <i>file</i> isn't specified, append to filename typescript in the current directory.
<i>file</i>	Write (or append) to <i>file</i> instead of filename typescript in the current directory.

NOTES

The **script** command places everything in the log file. This is not what the naive user expects.

NAME

scrto - set/show screen timeout

SYNOPSIS

scrto [**-none**] [*n*]

DESCRIPTION

scrto sets or displays the number of minutes the system waits before it shuts off the display screen. It begins counting minutes after the last input event or window configuration change.

By default, the system waits 15 minutes before it shuts off the display. Domain/OS turns the display back on whenever it receives an input event from the keyboard or mouse or whenever the DM creates, pops, moves, or resizes a window.

n (optional) Set the number of Domain/OS minutes for to wait before it shuts off the display.

Default if omitted: display current timeout setting

OPTIONS

-none Disables automatic timeout; never turn off the display.

EXAMPLES

Shows initial setting.

```
$ scrto
```

```
The screen timeout is set to 15 minutes
```

```
$
```

Sets delay to 10 min.

```
$ scrto 10
```

```
$
```

NAME

sed – stream editor

SYNOPSIS

sed [**-n**] [**-e script**] [**-f sfile**] [*file* ...]

DESCRIPTION

The stream editor **sed** copies the named *files* (standard input is the default) to the standard output, edited according to a script of commands.

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.

OPTIONS

- f sfile** Take the script of commands from the file *sfile*; these options accumulate.
- e script** Take the following argument as a command script. If there is just one **-e** option and no **-foptions**, you can omit the flag **-e**.
- n** Suppress the default output; copy only those lines specified by a **p** (print) function to the standard output.

ADDRESSES

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, */regular expression/*, in the style of **ed(1)** modified thus:

The escape sequence **\n** matches a newline embedded in 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 non-selected pattern spaces by use of the negation function **!** (see below).

FUNCTIONS

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

An argument denoted *text* consists of one or more lines, all but the last of which end with a \ (backslash) to hide the newline. Backslashes in *text* are treated like backslashes in the replacement string of an *s* command, and can be used to protect initial blanks and tabs against the stripping that is done on every script line.

An argument denoted *rfile* or *wfile* 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 Append. Place *text* on the output before reading the next input line.
- (2) **b label** Branch to the ':' command bearing the *label*. If *label* is empty, branch to the end of the script.
- (2) **c**
text Change. Delete the pattern space. With 0 or 1 address or at the end of a 2-address range, place *text* on the output. Start the next cycle.
- (2) **d** Delete the pattern space. Start the next cycle.
- (2) **D** Delete the initial segment of the pattern space through the first newline. Start the next cycle.
- (2) **g** Replace the contents of the pattern space by the contents of the hold space.
- (2) **G** Append the contents of the hold space to the pattern space.
- (2) **h** Replace the contents of the hold space by the contents of the pattern space.
- (2) **H** Append the contents of the pattern space to the hold space.
- (1) **i**
text Insert. Place *text* on the standard output.
- (2) **n** Copy the pattern space to the standard output. Replace the pattern space with the next line of input.
- (2) **N** Append the next line of input to the pattern space with an embedded newline. (The current line number changes.)
- (2) **p** Print. Copy the pattern space to the standard output.
- (2) **P** Copy the initial segment of the pattern space through the first newline to the standard output.
- (1) **q** Quit. Branch to the end of the script. Do not start a new cycle.
- (2) **r rfile** Read the contents of *rfile*. Place them on the output before reading the next input line.

(2) *s/regular expression/replacement/flags*

Substitute the *replacement* string for instances of the *regular expression* in the pattern space. Any character can be used instead of '/'. For a fuller description see *ed(1)*. *flags* is zero or more of

g Global. Substitute for all nonoverlapping instances of the *regular expression* rather than just the first one.

p Print the pattern space if a replacement was made.

w wfile Write. Append the pattern space to *wfile* if a replacement was made.

(2) *t label*

Test. Branch to the : command bearing the *label* if any substitutions have been made since the most recent reading of an input line or execution of a *t*. If *label* is empty, branch to the end of the script.

(2) *w wfile*

Write. Append the pattern space to *wfile*.

(2) *x*

Exchange the contents of the pattern and hold spaces.

(2) *y/string1/string2/*

Transform. Replace all occurrences of characters in *string1* with the corresponding character in *string2*. The lengths of *string1* and *string2* must be equal.

(2) *! function*

Don't. Apply 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) *=*

Place the current line number on the standard output as a line.

(2) *{*

Execute the following commands through a matching '*}*' only when the pattern space is selected.

(0)

An empty command is ignored.

SEE ALSO

ed(1), *grep(1)*, *awk(1)*, *lex(1)*

NAME

sh – the Bourne shell command language

SYNOPSIS

sh [**-ceiknrstuvx**] [**-Dname=value ...**] [*arg*] ...

DESCRIPTION

sh is a command programming language that executes commands read from a terminal or a file. 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 0; refer to `execve(2)` for further details. The *value* of a simple command is its exit status if it terminates normally, or 200+*status* if it terminates abnormally. See `sigvec(2)` for a list of status values.

A *pipeline* is a sequence of one or more *commands* separated by `|`. The standard output of each command but the last is connected by a `pipe(2)` 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 `&`. `;` and `&` have equal precedence which is lower than that of `&&` and `||`, `&&` and `||` also have equal precedence. A semicolon causes sequential execution; an ampersand causes the preceding pipeline to be executed without waiting for it to finish. The symbol `&&` (`||`) causes the *list* following to be executed only if the preceding *pipeline* returns a zero (non zero) value. Newlines may appear in a *list*, instead of semicolons, to delimit commands.

A *command* is either a simple-command or one of the following. The value returned by a command is that of the last simple-command executed in the command.

COMMANDS

for *name* [**in** *word ...*] **do** *list* **done**

Set *name* to the next word in the **for** word list each time a **for** command is executed. If **in** *word ...* is omitted, **in** `$@` is assumed. Execution ends when there are no more words in the list.

case *word* **in** [*pattern* [*|pattern ...*] *list* ;;] ... **esac**

Execute the *list* associated with the first pattern that matches *word*. The form of the patterns is the same as that used for filename generation.

if *list* **then** *list* [**elif** *list* **then** *list*] ... [**else** *list*] **fi**

Execute the *list* following **if**, and if it returns zero, execute the *list* following **then**. Otherwise, execute the *list* following **elif**, and if its value is zero, execute the *list* following **then**. Failing that, execute the **else** *list*.

while list [do list] done

Repeatedly execute the **while list**, and if its value is zero, execute the **do list**; otherwise, terminate the loop. The value returned by a **while** command is that of the last executed command in the **do list**. To negate the loop termination test, use **until** in place of **while**.

(list) Execute *list* in a sub-shell.

{ list } Simply execute *list*.

sh recognizes the following words only when they are the first word of a command, and only when they are not quoted:

if then else elif fi case in esac for while until do done { }

COMMAND SUBSTITUTION

Use the standard output from a command, enclosed in a pair of back quotes (` `), as part or all of a word. Trailing newlines are removed.

PARAMETER SUBSTITUTION

A *parameter* is a sequence of letters, digits, or underscores (a *name*), a digit, or any of the following characters: * @ # ? - \$!. 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] ..

\${parameter}

Substitute the value, if any, of the parameter. 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. If *parameter* is a digit, it is a positional parameter. If *parameter* is * or @ then all the positional parameters, starting with \$1, are substituted separated by spaces. \$0 is set from argument zero when the shell is invoked.

\${parameter-word}

If *parameter* is set, substitute its value; otherwise, substitute *word*.

\${parameter=word}

If *parameter* is not set, set it to *word*; then, substitute the value of the parameter. You may not assign positional parameters in this manner.

\${parameter?word}

If *parameter* is set, substitute its value; otherwise, print *word* and exit from the shell. If *word* is omitted, a standard message is printed.

\${parameter+word}

If *parameter* is set, substitute *word*; otherwise, substitute nothing.

In the above, **sh** does not evaluate *word* unless it is to be used as the substituted string. Thus, for example, **echo \${d-`pwd`}** only executes **pwd** if *d* is unset.

The following *parameters* are automatically set by the shell:

Number of positional parameters in decimal.

- Options supplied to the shell on invocation or by **set**.
- ? Value returned by the last executed command in decimal.
- \$ Process number of this shell.
- ! Process number of the last background command invoked.

The following *parameters* are used but not set by the shell:

- HOME** Default argument (home directory) for the **cd** command.
- PATH** The search path for commands (see EXECUTION).
- SHENV** If this parameter is set, the shell performs parameter substitution on the value to generate the pathname of the startup script containing commands that the shell executes every time a new shell is invoked. No error results if the file specified by the **SHENV** parameter doesn't exist or can't be read.
- MAIL** If this variable is set to the name of a mail file, the shell informs you of the arrival of mail in the specified file.
- PS1** Primary prompt string; this is a dollar sign (\$) by default.
- PS2** Secondary prompt string; this is a greater-than sign (>) by default.
- IFS** Internal field separators; these usually include *space*, *tab*, and *new-line*. **IFS** is ignored if **sh** is running as root or if the effective user id differs from the real user id.

BLANK INTERPRETATION

After parameter and command substitution, **sh** scans the subsequent results for internal field separator characters (those found in **\$IFS**), and then splits its output 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, **sh** scans each command word for the characters *, ?, and [. If one of these characters appears, 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, the word is left unchanged. The character . at the start of a filename or immediately following a /, and the character /, must be matched explicitly.

- * Matches any string, including the null string.
- ? Matches any single character.
- [...] Matches any one of the characters enclosed. A pair of characters separated by — matches any character lexically between the pair.

QUOTING

The following characters have a special meaning to the shell and cause termination of a word unless quoted:

; & () | < > newline space tab

You may quote a character by preceding it with a \ **sh** ignores a \newline. All characters enclosed between a pair of single quotation marks (``), except a single quote, are quoted. Parameter and command substitution occurs inside double quotes (""). A \ quotes the following characters: ` ` " and \$.

A "\$*" is equivalent to "\$1 \$2 ...", while "\$@" is equivalent to "\$1" "\$2"....

PROMPTING

When used interactively, **sh** prompts with the value of **PS1** before reading a command. If, at any time, you type a newline and need further input to complete a command, **sh** issues the secondary prompt (**PS2**).

INPUT/OUTPUT

Before you execute a command, you may redirect its output by using a special notation interpreted by the shell. The following may appear anywhere in a simple command or may precede or follow a *command*, and 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, create it; otherwise, truncate it to zero length.

>>*word* Use file *word* as standard output. If the file exists, append output (by seeking to the end); otherwise, create the file.

<<*word* Read the shell input up to a line the same as *word* or end-of-file. Make the resulting document the standard input. If any character of *word* is quoted, place no interpretation upon the characters of the document. Otherwise, do parameter and command substitution, ignore **newline** and use \ to quote the following characters: \ \$ ' and the first character of *word*.

< &*digit*

Duplicate the standard input from file descriptor *digit*. This works similarly for the standard output using >. Refer to **dup(2)** for further details.

< & - Close the standard input. Perform the same function on the standard output, using >.

If one of the above is preceded by a digit, the file descriptor created is that specified by the digit (instead of the default 0 or 1). For example,

```
... 2>&1
```

creates file descriptor 2 to be a duplicate of file descriptor 1.

If a command is followed by & 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 is a list of name-value pairs that is passed to an executed program in the same way as a normal argument list. Refer to **execve(2)** and **environ(7)**. 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 you modify the values of these parameters, or create new ones, the environment is unaffected, unless you use the **export** command 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.

You may augment the environment for any simple command by prefixing it with one or more assignments to parameters. Thus, the following two lines are equivalent:

```
TERM=450 cmd args
(export TERM; TERM=450; cmd args)
```

SIGNALS

sh ignores the **INTERRUPT** and **QUIT** signals for an invoked command if the command is followed by **&**. Otherwise, signals have the values inherited by the shell from its parent (see also **trap**).

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 an **execve(2)**.

The **\$PATH** shell parameter defines the search path for the directory containing the command. Each alternative directory name is separated by a colon (:). The default path is **/usr/ucb/bin:/usr/bin:/usr/apollo/bin**. If the command name contains a **/**, **sh** does not use the search path. Otherwise, **sh** searches each directory in the path 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 sub-shell (i.e., a separate process) is spawned to read it. A command in parentheses is also executed in a sub-shell.

SPECIAL COMMANDS

The following commands are executed in the shell process, and except where specified, no input/output redirection is permitted for such commands.

- #** For non-interactive shells, treat everything following the **#** as a comment, i.e., ignore the rest of the line. For interactive shells, the **#** has no special effect.
- :** No effect; this command does nothing. A zero exit code is returned.
- . file** Read and execute commands from *file* and return. Use the search path **\$PATH** to find the directory containing *file*.
- break [n]**
Exit from the enclosing **for** or **while** loop, if any. If *n* is specified, break *n* levels.
- continue [n]**
Resume the next iteration of the enclosing **for** or **while** loop. If *n* is specified, resume at the *n*th enclosing loop.
- cd [arg]**
Change the current directory to *arg*. The **\$HOME** shell parameter is the default *arg*.
- eval [arg ...]**
Read the arguments as input to the shell and execute the resulting command(s).

exec [*arg* ...]

Execute the command specified by the arguments in place of this shell without creating a new process. You may supply input/output arguments, and if you give no other type of arguments, **sh** modifies the input/output.

exit [*n*]

If the shell is not interactive, exit with the exit status specified by *n*. If you omit *n*, the exit status is that of the last command executed. (An end-of-file will also exit from the shell.)

export [*name* ...]

Mark the given names for automatic export to the *environment* of subsequently-executed commands. If no arguments are given, print a list of exportable names.

login [*arg* ...]

Equivalent to an **exec login arg ...** command.

read *name* ...

Read one line from the standard input. Assign successive words of the input to the variables *name* in order, with leftover words to the last variable. The return code is 0 unless the end-of-file is encountered.

readonly [*name* ...]

Mark the given names as read-only. You cannot change the values of these names by subsequent assignment. If no arguments are given, print a list of all read-only names.

rootnode [*arg*]

Change the current node entry directory to *arg*.

set [-ekntuvx [*arg* ...]]

- e Exit immediately if a command fails (on some systems, this switch works only on shells that are not interactive).
- k Place all keyword arguments in the environment for a command, not just those that precede the command name. The following, used in the shell, prints *a=b c* and *c*:

```
echo a=b c
set -k
echo a=b c
```

- n Read commands but do not execute them.
- t Exit after reading and executing one command.
- u Treat unset variables as an error when substituting.
- v Print shell input lines as they are read.
- x Print commands and their arguments as they are executed.
- Turn off the -x and -v options.

You can also use these options when you first invoke the shell. The current set may be found in `$-`.

Remaining arguments are positional parameters. The shell assigns them, in order, to `$1`, `$2`, etc. If no arguments are given, it prints the values of all names.

shift The positional parameters from `$2...` are renamed `$1...`

times Print the accumulated user and system times for processes run from the shell.

trap [*arg*] [*n*] ...

Read and execute *arg* upon receipt of signal(s) *n*. (Scan *arg* once when the trap is set and once when the trap is taken.) Execute **trap** commands in order of signal number. If *arg* is absent, reset all **trap**(s) *n* to their original values. If *arg* is the null string, the shell and invoked commands ignore this signal. If *n* is 0, execute the command *arg* on exit from the shell. Otherwise, execute the command upon receipt of signal *n* as numbered in `sigvec(2)`. With no arguments, **trap** prints a list of commands associated with each signal number.

umask [*nnn*]

Set the user file creation mask to the octal value *nnn*. See `umask(2)` for details. If *nnn* is omitted, print the current value of the mask.

ver

ver *systype*

With no arguments, return the current value of the SYSTYPE environment variable that specifies the version of UNIX commands being executed by the shell. With a *systype* argument, change the SYSTYPE environment variable to either `bsd4.3` or `sys5.3`, depending on which is specified.

wait [*n*]

Wait for the specified process and report its termination status. If *n* is not given, wait for all currently active child processes. The return code from this command is that of the process being awaited.

inlib *pathname*

Install a user-supplied library specified by *pathname* in the current (shell) process. The library is used to resolve external references of programs (and libraries) loaded after its installation. Note that the library is not loaded into the address space unless it is needed to resolve an external reference. The list of inlibed libraries is passed to all children of the current shell. Use **llib(1)** to examine this list.

COMMAND LINE OPTIONS

If the first character of argument zero is **-**, **sh** reads commands from **\$HOME/.profile**, if such a file exists. It then reads commands as described below. The following options are interpreted by the shell when it is invoked.

- c *string*** Read commands from *string*.
- s** Read commands from the standard input. Write shell output to file descriptor 2. (Note that the same activity occurs if no arguments remain.)
- i** Make the shell *interactive*. (Note that this also occurs if the shell input and output are attached to a terminal, as told by *gty*.) Ignore the terminate signal SIGTERM, so that **kill 0** does not kill the interactive shell. Catch and ignore the interrupt signal SIGINT, so that **wait** is interruptible. In all cases, ignore SIGQUIT.
- Dname=value** Use the **-D** option to specify a parameter *name*, that will be set to *value*, then passed into the shell's environment. This Domain/OS BSD option is useful for tailoring the environment of a shell invoked from a program that is not another shell (such as the Display Manager). If you set the SHENV parameter using this option, the startup script it specifies will be run. For example, if you define your <SHELL> key as follows: **cp /bin/sh -DSHENV=~/.shrc.pad**, the **~/.shrc.pad** script can contain commands to perform special processing for the pad and the shell. You can specify more than one **-D** option.

The remaining flags and arguments are described under the **set** command.

FILES

\$HOME/.profile
/tmp/sh*
/dev/null

DIAGNOSTICS

Errors detected by the shell, such as those that occur in syntax, cause **sh** to return a nonzero exit status. If the shell is not being used interactively, then execution of the shell file is abandoned. Otherwise, the exit status of the last command executed is returned. Refer to **exit** under SPECIAL COMMANDS for more information.

BUGS

If **<<** is used to provide standard input to an asynchronous process invoked by **&**, the shell gets confused about naming the input document. It creates a garbage file called **/tmp/sh*** and complains about not being able to find the file by another name.

SEE ALSO

csh(1), **ksh(1)**, **rootnode(1)**, **test(1)**, **execve(2)**, **environ(7)**

NAME

size – size of an object file

SYNOPSIS

size [*object* ...]

DESCRIPTION

size prints the (decimal) number of bytes required by the text, data, and bss portions, and their sum in hex and decimal, of each *object* file argument. If you don't specify a file, **size** uses **a.out**.

SEE ALSO

a.out(5)

NAME

sleep – suspend execution for an interval

SYNOPSIS

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

setitimer(2), alarm(3C), sleep(3)

BUGS

The value of *time* must be less than 2,147,483,647 seconds.

NAME

soelim – eliminate .so's from nroff input

SYNOPSIS

soelim [*file ...*]

DESCRIPTION

soelim reads the specified files or the standard input, and performs the textual inclusion implied by the **nroff** directives of the form

.so *somefile*

when they appear at the beginning of input lines. This is useful since programs such as **tbl** don't normally do this; it allows the placement of individual tables in separate files to be run as a part of a large document.

An argument consisting of a single dash (–) is taken to be a filename corresponding to the standard input.

Note that inclusion can be suppressed by using a backquote (`) instead of a period (.), as shown in the following example:

`so /usr/lib/tmac.s

EXAMPLE

A sample usage of **soelim** is:

soelim exum?.n | tbl | nroff –ms | col | lpr

BUGS

The format of the source commands must be standard. Exactly one blank must precede and no blanks follow the filename.

SEE ALSO

colcrt(1), **more(1)**

NAME

sort – sort or merge files

SYNOPSIS

sort [*options*] [*files*]

DESCRIPTION

sort sorts lines of all the named files together and writes the result on the standard output. A dash (–) used as a filename means the standard input. If you specify no input files, **sort** uses the standard input.

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.

OPTIONS

- b** Ignore leading blanks (spaces and tabs) in field comparisons.
 - d** Use dictionary order, making only letters, digits, and blanks significant in comparisons.
 - f** Fold uppercase letters onto lowercase.
 - i** Ignore characters outside the ASCII range 040-0176 in non-numeric comparisons.
 - n** Sort an initial numeric string (consisting of optional blanks, optional minus sign, and zero or more digits with optional decimal point) by arithmetic value. This option implies option **–b**.
 - r** Reverse the sense of comparisons.
 - tx** Specify *x* as the tab character separating fields.
 - +pos1 [–pos2]** Restrict a sort key to a field beginning at *pos1* and ending just before *pos2*. Both *pos1* and *pos2* have the form *m.n*, optionally followed by one or more of the flags **–bdfnr**, where *m* specifies the number of fields to skip from the beginning of the line and *n* specifies the 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 non-empty non-blank 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.

- c** Ensure that the input file is sorted according to the ordering rules; give no output unless the file is out of sort.
- m** Merge only; the input files are already sorted.
- o *name*** Use *name* as the output file instead of the standard output. This file may be the same as one of the inputs.
- T *directory*** Use *directory* as the name of a directory in which temporary files should be made.
- u** Suppress all but one line in each set of equal lines. Ignored bytes and bytes outside keys do not participate in this comparison.

EXAMPLES

To print in alphabetical order all the unique spellings in a list of words, with capitalized words distinguished from uncapitalized, use

```
sort -u +0f +0 list
```

To print the password file sorted by user ID number (the third colon-separated field as described in `passwd(5)`), use

```
sort -t: +2n /etc/passwd
```

To print the first instance of each month in an already sorted file of month/day entries, use the command line shown below. 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*`, `/tmp/*` first and second tries for temporary files

DIAGNOSTICS

Comments and exits with nonzero status for various trouble conditions and for disorder discovered under option `-c`.

BUGS

Very long lines are silently truncated.

SEE ALSO

`uniq(1)`, `comm(1)`, `rev(1)`, `join(1)`

NAME

sortbib – sort bibliographic database

SYNOPSIS

sortbib [*-sKEYS*] *database*

DESCRIPTION

sortbib sorts files of records containing **refer**(1) key-letters by user-specified keys. Records may be separated by blank lines, or by “.” and “.” delimiters, but the two styles may not be mixed together. This program reads through each *database* and pulls out key fields, which are sorted separately. The sorted key fields contain the file pointer, byte offset, and length of corresponding records. These records are delivered using disk seeks and reads, so **sortbib** may not be used in a pipeline to read standard input.

By default, **sortbib** alphabetizes by the first %A and the %D fields, which contain the senior author and date. The *-s* option is used to specify new *KEYS*. For instance, *-sATD* will sort by author, title, and date, while *-sA+D* will sort by all authors and by date.

sortbib sorts on the last word on the %A line, which is assumed to be the author's last name. A word in the final position, such as “jr.” or “ed.”, will be ignored if the name beforehand ends with a comma. Authors with two-word last names or unusual constructions can be sorted correctly by using the **nroff**(1) convention “\0” in place of a blank. A %Q field is considered to be the same as %A, except sorting begins with the first, not the last, word.

sortbib sorts on the last word of the %D line, usually the year. It also ignores leading articles (of any modern European language) when sorting by titles in the %T or %J field. If a sort-significant field is absent from a record, **sortbib** places that record before other records containing that field.

SEE ALSO

refer(1), **addbib**(1), **roffbib**(1), **indxbib**(1), **lookbib**(1)

BUGS

Records with missing author fields should probably be sorted by title.

Sort keys past the fourth one specified are not meaningful.

No more than 16 databases may be sorted together at one time. Records longer than 4096 characters will be truncated.

NAME

spell, spellin, spellout – find spelling errors

SYNOPSIS

spell [-v] [-b] [-x] [-d *hlist*] [-s *hstop*] [-h *spellhist*] [*file*]

spellin [*list*]

spellout [-d] *list*

DESCRIPTION

spell collects words from the named documents and looks them up in a spelling list. It prints on the standard output all words which neither occur among nor are derivable (by applying certain inflections, prefixes or suffixes) from words in the spelling list. If you do not name any files, **spell** collects words from the standard input.

spell ignores most **troff**(1), **tbl**(1), and **eqn**(1) constructions.

The spelling list is based on many sources. While it is more haphazard than an ordinary dictionary, it is also more effective with proper names and popular technical words. Coverage of the specialized vocabularies of biology, medicine and chemistry is light.

The history file accumulates copies of all output. The stop list filters out misspellings (e.g., *thier=thy-y+ier*) that would otherwise pass.

Two routines help maintain the hash lists used by **spell**. Both expect a set of words, one per line, from the standard input. **spellin** combines the words from the standard input and the preexisting *list* file and places a new list on the standard output. If no *list* file is specified, the new list is created from scratch.

spellout looks up each word from the standard input and prints on the standard output those that are missing from (or present on, with option -d) the hashed *list* file.

OPTIONS

- v Print all words not literally in the spelling list, and indicate plausible derivations from spelling list words.
- b Check British spelling. Besides preferring “centre”, “colour”, “speciality”, and “travelled”, this option insists upon “-ise” in words like “standardise”.
- x Print every plausible stem with an equal sign (=) for each word.
- d *hlist* Specify *hlist* as the spelling list file to be used instead of the default.
- s *hstop* Specify *hstop* as the stop list file to be used instead of the default.
- h *spellhist* Specify *spellhist* as the history file to be used instead of the default.

EXAMPLE

To verify that "hookey" is not on the default spelling list, add it to your own private list, and then use it with **spell**, use

```
echo hookey | spellout /usr/dict/hlista
echo hookey | spellin /usr/dict/hlista >
spell -d myhlist huckfinn
```

FILES

/usr/dict/hlist{ab}	Hashed spelling lists, American and British; default for -d
/usr/dict/hstop	Hashed stop list; default for -s
/dev/null	History file; default for -h
/tmp/spell.\$\$*	Temporary files
/usr/lib/spell	

BUGS

The spelling list's coverage is uneven; new installations will probably wish to monitor the output for several months to gather local additions.

British spelling was done by an American.

SEE ALSO

deroff(1), sort(1), tee(1), sed(1)

NAME

spline – interpolate smooth curve

SYNOPSIS

spline [*option*] ...

DESCRIPTION

spline takes pairs of numbers from the standard input as abscissas and ordinates of a function. It produces a similar set, which is approximately equally spaced and includes the input set, on the standard output. The cubic spline output (R. W. Hamming, *Numerical Methods for Scientists and Engineers*, 2nd ed., 349ff) has two continuous derivatives, and sufficiently many points to look smooth when plotted, for example by **graph**(1G).

OPTIONS

spline recognizes the following options, each as a separate argument.

- a [*space*] Supply abscissas automatically (they are missing from the input); spacing is given by the next argument, or is assumed to be 1 if the next argument is not a number.
- k [*n*] The constant *k* used in the boundary value computation
 $y_0'' = ky_1'', y_n'' = ky_{n-1}''$
 is set by the next argument. By default *k* = 0.
- n Space output points so that approximately *n* intervals occur between the lower and upper *x* limits. (Default *n* = 100.)
- p Make output periodic, that is, match derivatives at ends. First and last input values should normally agree.
- x [*l* [*u*]] Next one (or two) arguments are lower (and upper) *x* limits. Normally these limits are calculated from the data. Automatic abscissas start at lower limit (default 0).

DIAGNOSTICS

When data is not strictly monotone in *x*, **spline** reproduces the input without interpolating extra points.

BUGS

A limit of 1000 input points is enforced silently.

SEE ALSO

graph(1G), **plot**(1G)

NAME

split – split a file into pieces

SYNOPSIS

split [*-n*] [*file* [*name*]]

DESCRIPTION

split reads *file* and writes it in *n*-line pieces (default 1000) into a set of output files. The name of the first output file is *name* with **aa** appended, and so on alphabetically with subsequent files.

If you give no output name, **x** is the default. If you specify no input file, or if a dash (–) is given instead, **split** uses the standard input.

NAME

start_csh – start a log-in C shell

SYNOPSIS

start_csh

DESCRIPTION

The **start_csh** command starts a log-in C shell (one that reads the files `~/login` and `~/cshrc` if such files exist).

NOTES

See **csh(1)** for more information about particular properties of the C shell.

When you log in, the first shell started is a log-in shell (i.e., the first argument is preceded by a `-`). Typing **start_csh** lets you start a “log-in” shell even after initial login. Executing **start_csh** more than once causes the log-in script `~/login` to run more than once, which could cause unexpected results. **start_csh** is supported for compatibility with previous releases, and is not recommended.

SEE ALSO

csh(1), **start_sh(1)**, **login_sh(8)**

NAME

start_sh – start a log-in Bourne shell

SYNOPSIS

start_sh

DESCRIPTION

The **start_sh** command starts a log-in Bourne shell, which reads the files **/etc/profile** and **\$HOME/.profile** (if such a file exists).

NOTES

See **sh(1)** for more information about particular properties of the Bourne shell.

When you log in, the first shell started is a log-in shell (i.e., the first argument is preceded by a **-**). **start_sh** lets you start a “log-in” shell even after initial login. Executing **start_sh** more than once causes the **/etc/profile** and **~/.profile** scripts to be run more than once, which could cause unexpected results. **start_sh** is supported for compatibility with previous releases, and is not recommended.

SEE ALSO

sh(1), **start_csh(1)**, **login_sh(8)**

STCODE(1)

Domain/OS BSD

STCODE(1)

NAME**stcode** – translate status code value to text message**SYNOPSIS****stcode** *hex_stat_code***DESCRIPTION**

stcode prints the text message associated with a hexadecimal status code. This command is useful when a user program produces a hexadecimal status code instead of the textual message.

stcode processes predefined status codes. No provision is currently made to add user-defined status codes to the error text database.

hex_stat_code (required)

Specifies hexadecimal status code to be translated.

EXAMPLES

```
$ stcode 80001
```

```
disk not ready (from OS / disk manager)
```

NAME

stk_font_gen – create a custom stroke font

SYNOPSIS

stk_font_gen [*options*]

DESCRIPTION

stk_font_gen lets you create custom stroke fonts by using a Graphic Services Routine (GSR) tool called the Stroke Font Generator. This tool lets you edit existing stroke font files and change any or all of the font's character definitions. The Stroke Font Generator creates Apollo ASCII font files from Apollo binary font files and vice versa. Once you have created an ASCII version of an existing stroke font, you can modify the character definitions in the font, thereby creating a new font.

ARGUMENTS

-aa *Apollo ASCII font file* (required)

The **-aa** option specifies Apollo ASCII font file format; the file following this option must be an Apollo ASCII font file.

Default if omitted: No font is converted.

-ab *Apollo ASCII font file* (required)

The **-ab** option specifies Apollo binary font file format; the file following this option must be an Apollo binary font file.

Default if omitted: No font is converted.

EXAMPLES

The following steps outline the procedure for altering an existing font file to create a new font file:

1. Use the **stk_font_gen** command to create an ASCII font file from an existing binary font file. The following command creates an ASCII font file named *new_ascii_font_file* from the existing binary font file *gsr.stk_font*:

```
$stk_font_gen -ab /sys/gsr/gsr.stk_font -aa /sys/gsr/new_ascii_font_file
```

2. Modify any of the character definitions in the ASCII font file. In this example we modify the file so that the character corresponding to *index 42* is a star centered around the character origin (in the ASCII collating sequence, this character would normally be an asterisk).

```
index 42
W 50
Ox 0 Oy 0
```

```
M
0 40
L
```

```

23 -38
-38 12
38 12
-23 -38
0 40

```

3. Create a binary font file from the modified ASCII font file. In the following example, the new binary font file is called *my_font_file*

```
$ stk_font_gen -aa /sys/gsr/new_asci_font_file -ab /sys/gsr/my_font_file
```

Once you have created the new font file, load it and make it current as in the following code segment:

```

gpr_$load_font_file( '<F11>/sys/gsr/my_font_file<F0>' )
sizeof(<F11>/sys/gsr/my_font_file<F0>)
font_id { Returned font identifier }
status
;

buffer[length + 0] := gsr_$txt_font_id;
buffer[length + 1] := 2;
buffer[length + 2] := 0;
buffer[length + 3] := font_id;

```

NAME

strings - find the printable strings in an object file

SYNOPSIS

strings [-] [-a] [-o] [-number] *file* . . .

DESCRIPTION

The **strings** command looks for ASCII strings in a binary file. A string is normally defined as any sequence of four or more printing characters ending with a newline or a null. **strings** is useful for identifying random object files.

OPTIONS

-
- a Unless the - or -a option is specified, **strings** only looks in the initialized data space of object files.
- o Print each string preceded by its offset from the base of the *file* (in octal).
- number Use *number*, rather than 4, as the minimum string length.

EXAMPLES

The following examples demonstrate typical usages of **strings**. The *file* in this case is */bin/nice*. The first example lists the strings; the second example lists the strings, along with their offset from the file base:

```
% strings /bin/nice
@(#) Copyright (c) 1980 Regents of the University of California.
All rights reserved.
@(#) nice.c
3.1 - 87/07/15
usage: nice [ -n ] command
setpriority
%

% strings -o /bin/nice
1024 @(#) Copyright (c) 1980 Regents of the University of California.
1089 All rights reserved.
1112 @(#) nice.c
1123 3.1 - 87/07/15
1140 usage: nice [ -n ] command
1168 setpriority
%
```

SEE ALSO

od(1)

NAME

strip – remove symbols and relocation bits

SYNOPSIS

strip *name* ...

DESCRIPTION

strip removes the symbol table and relocation bits ordinarily attached to the output of the compiler and linker. This is useful for saving space after a program has been debugged.

The effect of **strip** is the same as use of the **-s** option of **ld**.

Domain/OS BSD OPTIONS

-Aa

Strip all debugging information, including that needed for trace-back. The **.blocks** and **.lines** sections, if present, will be removed. This option strips more information than the default strip behavior, and is added for users who wish to remove all symbolic information.

FILES

/tmp/stm? temporary file

SEE ALSO

ld(1)

NAME

struct – structure FORTRAN programs

SYNOPSIS

struct [*options*] [*file*]

DESCRIPTION

struct translates the FORTRAN program specified by *file* into a Ratfor program. (If no filename is given, **struct** reads the standard input.) Wherever possible, Ratfor control constructs replace the original FORTRAN. Statement numbers appear only where still necessary. **struct** makes certain cosmetic changes, including changing Hollerith strings into quoted strings and relational operators into symbols (e.g., “.GT.” into “>”). The output is appropriately indented.

OPTIONS

- s Accept input in standard format: comments are specified by a c, C, or * in column 1, and continuation lines are specified by a nonzero, nonblank character in column 6. Normally input is in the form accepted by **f77**(1).
- i Do not turn computed **goto** statements into switches. (Ratfor does not turn switches back into computed **goto** statements.)
- a Turn sequences of **else-ifs** into a non-Ratfor switch of the form

```
switch
{
    case pred1: code
    case pred2: code
    case pred3: code
    default: code
}
```

The case predicates are tested in order; the code appropriate to only one case is executed. This generalized form of switch statement does not occur in Ratfor.

- b Generate **goto**’s instead of multilevel break statements.
- n Generate **goto**’s instead of multilevel next statements.
- tn Make the nonzero integer *n* the lowest valued label in the output program (default 10).
- cn Increment successive labels in the output program by the nonzero integer *n* (default 1).
- en If *n* is 0 (default), place code within a loop only if it can lead to an iteration of the loop. If *n* is nonzero, admit a small code segment to a loop if otherwise the loop would have exits to several places including the segment and the segment can be reached only from the loop. “Small” is close to, but not equal to, the number of statements in the code segment. Values of *n* under 10 are suggested.

FILES

/tmp/struct*
/usr/lib/struct/*

BUGS

Struct knows FORTRAN 66 syntax, but not full FORTRAN 77.

If an input FORTRAN program contains identifiers which are reserved words in Ratfor, the structured version of the program will not be a valid Ratfor program.

The labels generated cannot go above 32767.

If you get a `goto` without a target, try using `-e`.

SEE ALSO

`f77(1)`, `ratfor(1)`

NAME

stty – set terminal options

SYNOPSIS

stty [*option ...*]

DESCRIPTION

stty sets certain I/O options on the current output terminal, placing its output on the diagnostic output. With no argument, it reports the speed of the terminal and the settings of the options which are different from their defaults. Use of one of the following options modifies the output as described:

OPTIONS

all	All normally used option settings are reported.
everything	Everything stty knows about is printed.
speed	The terminal speed alone is printed on the standard output.
size	The terminal (window) sizes are printed on the standard output, first rows and then columns.
The option strings are selected from the following set:	
even	Allow even parity input.
–even	Disallow even parity input.
odd	Allow odd parity input.
–odd	Disallow odd parity input.
raw	Raw mode input (no input processing (erase, kill, interrupt, ...); parity bit passed back).
–raw	Negate raw mode.
cooked	Same as –raw .
cbreak	Make each character available to read(2) as received; no erase and kill processing, but all other processing (interrupt, suspend, ...) is performed.
–cbreak	Make characters available to read Only when newline is received.
–nl	Allow carriage return for newline, and output CR-LF for carriage return or newline.
nl	Accept only newline to end lines.
echo	Echo back every character typed.
–echo	Do not echo characters.
lcase	Map uppercase to lowercase.
–lcase	Do not map case
tandem	Enable flow control, so that the system sends out the stop character when its internal queue is in danger of overflowing on input, and sends the start character when it is ready to accept further input.
–tandem	Disable flow control.
–tabs	Replace tabs by spaces when printing.

tabs Preserve tabs.
ek Set erase and kill characters to # and @.

For the following commands which take a character argument *c*, you may also specify *c* as the "u" or "undef", to set the value to be undefined. A value of "x", a two character sequence, is also interpreted as a control character, with "?" representing delete.

erase *c* Set erase character to *c* (default '#', but often reset to CTRL/H).
kill *c* Set kill character to *c* (default '@', but often reset to CTRL/U).
intr *c* Set interrupt character to *c* (default DEL or CTRL/? (delete), but often reset to CTRL/C).
quit *c* Set quit character to *c* (default CTRL/^\).
start *c* Set start character to *c* (default CTRL/Q).
stop *c* Set stop character to *c* (default CTRL/S).
eof *c* Set end of file character to *c* (default CTRL/D).
brk *c* Set break character to *c* (default undefined). This character is an additional character causing wakeup.

cr0 cr1 cr2 cr3 Select style of delay for carriage return (see **loctl(2)**).

nl0 nl1 nl2 nl3 Select style of delay for linefeed.

tab0 tab1 tab2 tab3 Select style of delay for tab.

ff0 ff1 Select style of delay for form feed.

bs0 bs1 Select style of delay for backspace.

tty33 Set all modes suitable for the Teletype Corporation Model 33 terminal.

tty37 Set all modes suitable for the Teletype Corporation Model 37 terminal.

vt05 Set all modes suitable for Digital Equipment Corp. VT05 terminal.

dec Set all modes suitable for Digital Equipment Corp. operating systems users; (erase, kill, and interrupt characters to CTRL/?, CTRL/U, and CTRL/C, decctlq and "newcr".)

tn300 Set all modes suitable for a General Electric TerminiNet 300.

ti700 Set all modes suitable for Texas Instruments 700 series terminal.

tek Set all modes suitable for Tektronix 4014 terminal.

0 Hang up phone line immediately.

50 75 110 134 150 200 300 600 1200 1800 2400 4800 9600 exta extb Set terminal baud rate to the number given, if possible. (These are the speeds supported by the DH-11 interface.)

rows *n* The terminal size is recorded as having *n* rows.

columns *n* The terminal size is recorded as having *n* columns.

cols *n* An alias for **columns**.

A teletype driver that supports the job control processing of `csch(1)` and more functionality than the basic driver is described in `tty(4)`. The following options apply only to it.

new	Use new driver (switching flushes typeahead).
crt	Set options for a CRT (<code>crtbs</code> , <code>ctlecho</code> and, if ≥ 1200 baud, <code>crterase</code> and <code>crtkill</code>).
crtbs	Echo backspaces on erase characters.
prterase	For printing terminal echo erased characters backwards within “\” and “/”.
crterase	Wipe out erased characters with “backspace-space-backspace.”
-crterase	Leave erased characters visible; just backspace.
crtkill	Wipe out input on line kill, ala <code>crterase</code> .
-crtkill	Just echo line kill character and a newline on line kill.
ctlecho	Echo control characters as “^x” (and delete as “^?”.) Print two backspaces following the EOT character (CTRL/D).
-ctlecho	Control characters echo as themselves; in cooked mode EOT (CTRL/D) is not echoed.
decctlq	After output is suspended (normally by CTRL/S), only a start character (normally CTRL/Q) will restart it. This is compatible with DEC’s vendor supplied systems.
-decctlq	After output is suspended, any character typed will restart it; the start character will restart output without providing any input. (This is the default.)
tostop	Background jobs stop if they attempt terminal output.
-tostop	Output from background jobs to the terminal is allowed.
tilde	Convert “~” to “^~” on output (for Hazeltine terminals).
-tilde	Leave poor “~” alone.
flusho	Output is being discarded usually because user hit CTRL/O (internal state bit).
-flusho	Output is not being discarded.
pendin	Input is pending after a switch from cbreak to cooked and will be re-input when a read becomes pending or more input arrives (internal state bit).
-pendin	Input is not pending.
pass8	Passes all 8 bits through on input, in any mode.
-pass8	Strips the 0200 bit on input except in raw mode.
mdmbuf	Start/stop output on carrier transitions (not implemented).
-mdmbuf	Return error if write attempted after carrier drops.
litout	Send output characters without any processing.
-litout	Do normal output processing, inserting delays, etc.

nohang	Don't send hangup signal if carrier drops.
—nohang	Send hangup signal to control process group when carrier drops.
etxack	Diablo style etx/ack handshaking (not implemented).

The following special characters are applicable only to the new teletype driver and are not normally changed.

susp <i>c</i>	Set suspend process character to <i>c</i> (default CTRL/Z).
dsusp <i>c</i>	Set delayed suspend process character to <i>c</i> (default CTRL/Y).
rprnt <i>c</i>	Set reprint line character to <i>c</i> (default CTRL/R).
flush <i>c</i>	Set flush output character to <i>c</i> (default CTRL/O).
werase <i>c</i>	Set word erase character to <i>c</i> (default CTRL/W).
lnext <i>c</i>	Set literal next character to <i>c</i> (default CTRL/V).

NOTES

Setting tty characters via **stty** will not affect Display Manager key bindings.

SEE ALSO

ioctl(2), tabs(1), tset(1), tty(4)

NAME

style – analyze surface characteristics of a document

SYNOPSIS

style [-mm] [-ml] [-a] [-e] [-l *num*] [-r *num*] [-p] [-P] *file*

DESCRIPTION

style analyzes the surface characteristics of a document's writing style. It reports on readability, sentence length and structure, word length and usage, verb type, and sentence openers. Because **style** runs **deroff(1)** before looking at the text, you should include formatting header files as part of the input.

OPTIONS

- mm Override the default macro package **ms(7)**.
- ml Cause **deroff(1)** to skip lists. This option should be used if the document contains many lists of non-sentences.
- a Print all sentences with their length (in words) and readability index.
- e Print all sentences that begin with an expletive.
- p Print all sentences that contain a passive verb.
- l *num* Print all sentences more than *num* words in length.
- r *num* Print all sentences whose readability index is greater than *num*.
- P Print parts of speech of the words in the document.

SEE ALSO

deroff(1), **diction(1)**

BUGS

Use of non-standard formatting macros may cause incorrect sentence breaks.

NAME

su – substitute user ID temporarily

SYNOPSIS

su [**-f**] [**-**] [*userid*]

DESCRIPTION

su permits the temporary substitution of a new user ID. Before allowing this change, however, it demands the password of the specified *userid*, and if it is given, **su** changes to that *userid* and invokes the appropriate shell without changing the current directory. The user environment is unchanged except for HOME, SHELL, and USER which are taken from the account information for the user being substituted (see *environ(7)*). The new user ID stays in force until the shell exits.

If you specify no *userid*, “root” is assumed. To remind the super users of their responsibilities, the shell substitutes a pound sign (#) for its usual prompt.

OPTIONS

- f** Prevent *csch(1)* from executing the *.cschrc* file, thus making **su** start up faster.
- Simulate a full login. This option must be followed by a space if specified with *userid*.

NOTES

You must belong to the group ID “0” to **su** to “root”.

SEE ALSO

sh(1), *csch(1)*, *passwd_override(5)*

NAME

sum – sum and count blocks in a file

SYNOPSIS

sum *file*

DESCRIPTION

sum calculates and prints a 16-bit checksum for the named *file*. It also prints the number of blocks in the *file*. **sum** is typically used to look for bad spots in a file, or to validate a file communicated over a transmission line.

DIAGNOSTICS

"Read error" is indistinguishable from end-of-file on most devices; check the block count.

SEE ALSO

wc(1)

NAME

swapul – rearrange underlining

SYNOPSIS

swapul

DESCRIPTION

swapul reads lines from standard input, rearranges underlining so that the underlines follow a character in the output stream (instead of being preceded by them), and writes the resulting text to standard output.

NAME

systype – display version stamp

SYNOPSIS

/usr/apollo/bin/systype *file*

DESCRIPTION

systype displays the UNIX version stamp of the specified object file. Four columns are displayed in the output. The first (OBJTYPE) contains the object type, the second (SYSTYPE) contains the **systype** (one of the following):

none
any
sys5
sys5.3
bsd4.2
bsd4.3

The third column (RUNTYPE) contains the **runtype**, which has the same set of possible types as the **systype**; and the forth column (FILE) is the name of the file.

The format is as follows:

OBJTYPE	SYSTYPE	RUNTYPE	FILE
coff	bsd4.3	sys5.3	/bin/cc
obj	bsd4.2		//node_1/bsd4.2/bin/cc

Note that files of object type OBJ, do not have runtypes.

EXAMPLES

\$ systype /bin/cc

OBJTYPE	SYSTYPE	RUNTYPE	FILE
coff	bsd4.3	sys5.3	/bin/cc

\$ systype /bsd4.2/bin/cc

OBJTYPE	SYSTYPE	RUNTYPE	FILE
obj	bsd4.2		/bsd4.2/bin/cc

SEE ALSO

cc(1), **ld(1)**;

Using Your BSD Environment

NAME

tabs – set terminal tabs

SYNOPSIS

tabs [**-n**] [*terminal*]

DESCRIPTION

tabs sets the tabs on a variety of terminals. It recognizes various terminal names given in **term**(7), but the default is suitable for most 300–baud terminals.

In most cases, it is more efficient and convenient to use **tset**(1) to set the tabs.

OPTIONS

-n Left margin is not indented.

BUGS

tabs does not affect Display Manager tab stops.

SEE ALSO

stty(1), **tset**(1), **term**(7)

NAME

tail – deliver the last part of a file

SYNOPSIS

tail [*±number*[*lbc*][*fr*]] [*file*]

DESCRIPTION

tail copies the named file to the standard output beginning at a designated place. If you don't name a file, **tail** uses the standard input.

Copying begins at distance *+number* from the beginning, or *-number* from the end of the input. *number* is counted in units of lines, blocks or characters, according to the appended option (see below). When no units are specified, counting is by lines.

OPTIONS

<i>+number</i>	Begin copying <i>number</i> of units from the beginning of the input.
<i>-number</i>	Begin copying <i>number</i> of units from the end of the input.
<i>±numberl</i>	Count in units of lines.
<i>±numberb</i>	Count in units of blocks.
<i>±numberc</i>	Count in units of characters.
<i>±numberr</i>	Print lines from the end of the file in reverse order. If you don't specify a <i>number</i> , the default for <i>r</i> is to print the entire file this way.
<i>±numberf</i>	Do not quit at the end of the file; wait and try to read repeatedly, hoping that the file will grow.

EXAMPLES

The following command will copy the last 10 characters of the file named **errfile** to the standard output.

tail -10c errfile

The following command prints the contents of the file **errfile** in reverse order.

tail -r errfile

BUGS

Tails relative to the end of the file are treasured in a buffer, and thus are limited in length.

Various kinds of anomalous behavior may happen with character special files.

SEE ALSO

dd(1), head(1)

NAME

talk – talk to another user

SYNOPSIS

talk *person* [*ttyname*]

DESCRIPTION

talk is a visual communication program that copies lines from your terminal to that of another user.

If you wish to talk to someone on your own machine, then *person* is just the person's log-in name. If you wish to talk to a user on another host, then *person* is of the form

```
host!user or
host.user or
host:user or
user@host
```

The form *user@host* is preferred.

If you want to talk to a user who is logged in more than once, use the *ttyname* argument to indicate the appropriate terminal name.

When first called, **talk** sends the following message to the user you wish to talk to:

```
Message from TalkDaemon@his_machine...
talk: connection requested by your_name@your_machine.
talk: respond with: talk your_name@your_machine
```

At this point, the recipient of the message should reply by typing

talk *your_name@your_machine*

It doesn't matter from which machine the recipient replies, as long as his log-in name is the same. Once communication is established, the two parties may type simultaneously, with their output appearing in separate windows. Typing CTRL/L causes the window to be reprinted, while your erase, kill, and word kill characters will work normally in **talk**. To exit, type your interrupt character; **talk** then moves the cursor to the bottom of the screen and restores the terminal.

Permission to talk may be denied or granted by use of the **mesg(1)** command. At the outset talking is allowed. Certain commands, in particular **nroff(1)** and **pr(1)**, disallow messages in order to prevent messy output.

For a node to accept incoming **talk** commands from remote machines, it must be correctly configured to run TCP/IP. See *Configuring and Managing TCP/IP* for information on configuring TCP/IP.

BUGS

The version of **talk(1)** released with 4.3BSD uses a protocol that is incompatible with the protocol used in the version released with 4.2BSD.

FILES

/etc/hosts	to find the recipient's machine
/etc/utmp	to find the recipient's tty

SEE ALSO

mesg(1), who(1), mail(1), write(1)

NAME

tar – tape archiver

SYNOPSIS

tar [*key*] [*name* ...]

DESCRIPTION

tar saves and restores multiple files on a single file (usually a magnetic tape, but it can be any file). The *key* argument is a string of characters containing at most one function letter and possibly one or more function modifiers. Other arguments to **tar** are file or directory names specifying which files to dump or restore. In all cases, appearance of a directory name refers to the files and (recursively) subdirectories of that directory.

FUNCTIONS

The function portion of the key is specified 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. This command implies **r**.
- r** Write the named files on the end of the tape. The **c** function implies this.
- x** Extract the named files from the tape. If the named file matches a directory whose contents have been written onto the tape, this directory is (recursively) extracted. The owner, modification time, and mode are restored (if possible). If no file argument is given, the entire content of the tape is extracted. Note that if multiple entries specifying the same file are on the tape, the last one overwrites all earlier copies. **tar** writes restored files to the current directory.
- t** The names of the specified 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** Add the named files to the tape only if they are not already there or are there, but have been modified since they were last archived. (Not supported by Apollo tape drivers.)

OPTIONS

The following characters may be used in addition to the letter that selects the functions desired.

- A** Include Apollo-specific information; this option allows **tar** to handle Domain/OS typed files.
- o** Suppress writing of owner and permission information from archived directories.
- p** Restore files to their original modes, ignoring the present **umask**(2). Restores setuid to the super-user.
- 0, ..., 9** Select an alternate drive on which the tape is mounted. The default is drive 0 at 1600 bpi, which is normally **/dev/rmt8**.
- v** Use verbose mode. Normally **tar** does its work silently. This option makes **tar** print the name of each file it treats preceded by the function

letter. With the **t** function, the verbose option gives more information about the tape entries than just their names.

- w** Print the action to be taken followed by filename, then wait for user confirmation. If you enter a word beginning with "y", **tar** does the action. Any other input means "no".
- f archive** Use *archive* as the name of the archive instead of */dev/rmt?*. If the name of the file is a dash (-), **tar** writes to standard output or reads from standard input as appropriate. Thus, **tar** can be used as the head or tail of a filter chain. **tar** can also be used to move hierarchies with the command
`cd fromdir; tar cf - . | (cd todir; tar xf -)`
- b factor** Use *factor* as the blocking factor for tape records. The default is 20. This option should be used only with raw magnetic tape archives. **tar** automatically determines block size when reading tapes (with functions **x** and **t**).
- l** Complain if **tar** cannot resolve all of the links to the files dumped.
- m** Do not restore the modification times. The modification time is set to the time of extraction.
- h** Force **tar** to follow symbolic links as if they were normal files or directories. Normally **tar** does not follow symbolic links.
- B** Force input and output blocking to 20 blocks per record. This option allows **tar** to work across a communications channel where the blocking may not be maintained.
- C dir** perform a **chdir(2)** to the filename preceded by this option. This option allows multiple directories not related by a close common parent to be archived using short relative path names. For example, to archive files from */usr/include* and from */etc*, you might use
`tar c -C /usr include -C`

Previous restrictions dealing with **tar**'s inability to properly handle blocked archives have been lifted.

EXAMPLES

tar cbf 1 /dev/rct8 /usr/you

Writes the contents of */usr/you* and all of its subdirectories to cartridge tape.

tar xf /dev/rct8

Restores all of the files from */dev/rct8* to the filesystem, writing them into the current working directory.

tar cf /tarfile /usr/you/src/*.c

Bundles all files in */usr/you/src* whose names end in ".c" into */tarfile*.

tar cfb 1 /dev/rct8 /usr/you

Won't work. Blocksize and archive file name must be in the same order as **b** and **f** option letters.

FILES

/dev/rmt?
/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.

NOTES

Cartridge tapes must be rewound with the command **mt rewind** prior to using the **tar** command.

BUGS

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 current limit on filename length is 100 characters.
There is no way to follow symbolic links selectively.
When you extract files created with the **r** or **u** options, directory modification times may not be set correctly.

SEE ALSO

mt(1), **rbak(1)**, **wbak(1)**, **rwmt(1)**, **tar(5)**

NAME

tb - print process traceback

SYNOPSIS

tb [*options*] [*process_spec*]

DESCRIPTION

tb prints a process traceback, listing the name and current line number of each routine on the call stack. There are two forms of traceback:

Active Traces the current state of an executing process.

Diagnostic Traces the state of an aborted process at the time of the fault which killed it.

process_spec (optional) UNIX process ID (PID), aegis process name, or aegis process UID. Process names are not recorded in the process dump file, so dead processes must be referenced by PID or UID. Since PID's are reused multiple dump file entries for the same PID are possible, the command will select the most recent.

Default if omitted: perform a diagnostic traceback for the last child of the invoking process

OPTIONS

- p[roc]** Traces exactly the specified process. If this option is absent, the specified process or one of its children may be traced, as described below.
- d[iagnostic]** Prints a diagnostic traceback of an aborted process.
- n[ode] *node_spec*** Uses the process dump file on the specified node. Implies **-diagnostic**.
- c[ommand] *pathname*** Prints diagnostic traceback(s) for processes running the specified program. *pathname* must be reachable from the working directory; command search rules are not applied. Implies **-diagnostic**.
- s[ince] *date_time_spec*** Prints diagnostic traceback(s) for processes which aborted after the specified time. Implies **-diagnostic**. The format for *date_time_spec* is `[[yyyy/]mm/dd][.][hh:mm[:ss]]`.
- l[ast] [*n*]** Prints the *n* most recent entries in the process dump file (which also satisfy other selection criteria if given). *n* defaults to 1. If neither **-last** nor **-all** is specified **tb** prints only the most recent entry. Implies **-diagnostic**.
- a[ll]** Prints all entries in the process dump file (which also satisfy other selection criteria if given.) If neither **-last** or **-all** are specified, **tb** prints only the most recent entry. Implies **-diagnostic**.

-f[ull]	Prints additional fault diagnostic information, such as register values. Implies -diagnostic .
-b[rief]	Lists entries in the process dump file that satisfy selection criteria, but do not print tracebacks. The listing shows the process, parent, and group IDs, the time of the dump, the abort status, and the program that was running.
-t[asks]	Traces all tasks in the process. By default only the currently active task is shown. Ignored if tasking is not enabled. Applies only to active process tracebacks.
-h[eaders_off]	Suppresses output of process ID, dump time, and program name preceding diagnostic traceback, or of column headers in brief format. It has no effect on active process traceback.

Diagnostic Tracebacks

A diagnostic traceback shows the state of the call stack at the time of a fault which causes a process to be aborted. Traceback information is written to **'node_data/system_logs/proc_dump'** at the time of the fault. This is a circular buffer in which the oldest information is overwritten as needed to make room for new. There is space for approximately 150-200 dumps. **tb** prints up to 128 call levels for diagnostic tracebacks.

tb prints a diagnostic traceback if the command line specifies **-diagnostic** or any option which implies it, or if the process specified is not active. If **-diagnostic** is specified together with an active process, the most recent aborted child of that process is traced (or most recent children if **-last** or **-all** is specified).

If no options are given (except possibly **-f**, **-b** or **-h**) **tb** prints a diagnostic traceback for the most recent aborted child of the process which invoked **tb**.

Examples of Requesting Diagnostic Tracebacks

Assume **process_5** is an active shell process, and process number 107 is not active. Traceback process 107.

```
$ tb 107
```

Traceback last aborted command invoked from **process_5**.

```
$tb -d process_5
```

Traceback last aborted command from this shell

```
$tb
```

Traceback last aborted process running **test3**

```
$tb -c test3
```

List all entries in the process dump file made today

```
$tb -s today -a -b
```

Active Process Tracebacks

An active process traceback shows the current state of an executing process, listing the name and line number of each procedure in the call stack. The process is suspended while the traceback is taken. **tb** prints an active process traceback if the command line specifies an active process and does not include **-diagnostic** (or any option that implies it). If the process is specified by name and has any active children, then the most recent child is traced. (This allows a process to be specified by the name of its invoking shell process.) This behavior may be overridden by the **-proc** switch, or by specifying the process by PID or UID. Note that the only other option applicable to active process tracebacks is **-task**.

Examples of Requesting Active Process Tracebacks

Assume **process_7** is an active shell process, from which a command running in process 747 has been invoked.

```
$tb 747
```

Traceback the invoked command

```
$tb process_7
```

same

```
$tb -p process_7
```

Traceback the shell process itself

NAME

tbl – format tables for nroff or troff

SYNOPSIS

tbl [*files*] ...

DESCRIPTION

tbl is a preprocessor that formats tables for **nroff** or **troff**(1). The input files are copied to the standard output, except for lines between **.TS** and **.TE** command lines, which are assumed to describe tables and are reformatted.

If you don't supply any arguments, **tbl** reads the standard input, so it may be used as a filter. When **tbl** is used with **eqn** or **neqn** the **tbl** command should be first, to minimize the volume of data passed through pipes. See *UNIX Text Processing* for more detailed information about **tbl**.

EXAMPLE

As an example, letting **\t** represent a tab (which should be typed as a genuine tab) the input

```
.TS
c s s
c c s
c c c
l n n.
Household Population
Town\tHouseholds
\tNumber\tSize
Bedminster\t789\t3.26
Bernards Twp.\t3087\t3.74
Bernardsville\t2018\t3.30
Bound Brook\t3425\t3.04
Branchburg\t1644\t3.49
Bridgewater\t7897\t3.81
Far Hills\t240\t3.19
.TE
```

yields the following table.

Household Population		
Town	Households	
	Number	Size
Bedminster	789	3.26
Bernards Twp.	3087	3.74
Bernardsville	2018	3.30
Bound Brook	3425	3.04
Branchburg	1644	3.49
Bridgewater	7897	3.81
Far Hills	240	3.19

SEE ALSO

troff(1), eqn(1), col(1), colcrt(1);
UNIX Text Processing;
M. E. Lesk, *TBL*.

NAME

tcopy – copy a mag tape

SYNOPSIS

tcopy *src* [*dest*]

DESCRIPTION

tcopy is designed to copy magnetic tapes. The only assumption made about the tape is that there are two tape marks at the end. **tcopy** with only a source specified will print information about the sizes of records and tape files. If a destination is specified, a copy will be made of the source tape. The blocking on the destination tape will be identical to that used on the source tape. Copying a tape produces the same screen output as merely printing the sizes.

SEE ALSO

mtio(4)

NAME

tee – pipe fitting

SYNOPSIS

tee [**-i**] [**-a**] [*files*] ...

DESCRIPTION

tee transcribes the standard input to the standard output and makes copies in the *files*.

OPTIONS

-i	Ignore interrupts.
-a	Append output to the named files rather than overwrite them.

NAME

telnet – user interface to the TELNET protocol

SYNOPSIS

telnet [*host* [*port*]]

DESCRIPTION

telnet is used to communicate with another host using the **TELNET** protocol. If **telnet** is invoked without arguments, it enters command mode, indicated by its prompt ("telnet>"). In this mode, it accepts and executes the commands listed below. If it is invoked with arguments, it performs an **open** command (see below) with those arguments.

Once a connection has been opened, **telnet** enters an input mode. The input mode entered will be either "character at a time" or "line by line" depending on what the remote system supports.

In "character at a time" mode, most text typed is immediately sent to the remote host for processing.

In "line by line" mode, all text is echoed locally, and (normally) only completed lines are sent to the remote host. The "local echo character" (initially "E") may be used to turn off and on the local echo (this would mostly be used to enter passwords without the password being echoed).

In either mode, if the **localchars** toggle is TRUE (the default in line mode; see below), the user's **quit**, **intr**, and **flush** characters are trapped locally, and sent as **TELNET** protocol sequences to the remote side. There are options (see **toggle autoflush** and **toggle autosynch** below) which cause this action to flush subsequent output to the terminal (until the remote host acknowledges the **TELNET** sequence) and flush previous terminal input (in the case of **quit** and **intr**).

While connected to a remote host, **telnet** command mode may be entered by typing the **telnet** "escape character" (initially "^"). When in command mode, the normal terminal editing conventions are available.

COMMANDS

The following commands are available. Only enough of each command to uniquely identify it need be typed (this is also true for arguments to the **mode**, **set**, **toggle**, and **display** commands).

- | | | |
|--------------|---|---|
| open | open <i>host</i> [<i>port</i>] | Open a connection to the named host. If no port number is specified, telnet will attempt to contact a TELNET server at the default port. The host specification may be either a host name (see hosts(5)) or an Internet address specified in "dot notation" (see inet(3N)). |
| close | | Close a TELNET session and return to command mode. |
| quit | | Close any open TELNET session and exit telnet . An end of file (in command mode) will also close a session and exit. |
| z | | Suspend telnet . This command only works when the user is using the cs(1) . |

mode type	Ask the remote host for permission to go into the requested mode. If the remote host is capable of entering that mode, the requested mode will be entered. <i>Type</i> is either line (for "line by line" mode) or character (for "character at a time" mode).
status	Show the current status of telnet . This includes the peer one is connected to, as well as the current mode.
display [argument...]	Displays all, or some, of the set and toggle values (see below).
? [command]	Get help. With no arguments, telnet prints a help summary. If a command is specified, telnet prints the help information for just that command.
send arguments	Sends one or more special character sequences to the remote host. The following are the arguments which may be specified (more than one argument may be specified at a time):
escape	Sends the current telnet escape character (initially "[").
synch	Sends the TELNET SYNCH sequence. This sequence causes the remote system to discard all previously typed (but not yet read) input. This sequence is sent as TCP urgent data (and may not work if the remote system is a 4.2 BSD system — if it doesn't work, a lower case "r" may be echoed on the terminal).
brk	Sends the TELNET BRK (Break) sequence, which may have significance to the remote system.
ip	Sends the TELNET IP (Interrupt Process) sequence, which should cause the remote system to abort the currently running process.
ao	Sends the TELNET AO (Abort Output) sequence, which should cause the remote system to flush all output from the remote system to the user's terminal.
ayt	Sends the TELNET AYT (Are You There) sequence, to which the remote system may or may not choose to respond.
ec	Sends the TELNET EC (Erase Character) sequence, which should cause the remote system to erase the last character entered.
el	Sends the TELNET EL (Erase Line) sequence, which should cause the remote system to erase the line currently being entered.
ga	Sends the TELNET GA (Go Ahead) sequence, which likely has no significance to the remote system.

- nop** Sends the TELNET NOP (No Operation) sequence.
- ?** Prints out help information for the **send** command.

set argument value

Set any one of a number of **telnet** variables to a specific value. The special value **off** turns off the function associated with the variable. The values of variables may be interrogated with the **display** command. The variables which may be specified are:

- echo** This is the value (initially “E”) which, when in “line by line” mode, toggles between doing local echoing of entered characters (for normal processing), and suppressing echoing of entered characters (for entering, say, a password).
- escape** This is the **telnet** escape character (initially “[”) which causes entry into **telnet** command mode (when connected to a remote system).

interrupt

If **telnet** is in **localchars** mode (see **toggle localchars** below) and the **interrupt** character is typed, a TELNET IP sequence (see **send ip** above) is sent to the remote host. The initial value for the interrupt character is taken to be the terminal’s **intr** character.

- quit** If **telnet** is in **localchars** mode (see **toggle localchars** below) and the **quit** character is typed, a TELNET BRK sequence (see **send brk** above) is sent to the remote host. The initial value for the quit character is taken to be the terminal’s **quit** character.

flushoutput

If **telnet** is in **localchars** mode (see **toggle localchars** below) and the **flushoutput** character is typed, a TELNET AO sequence (see **send ao** above) is sent to the remote host. The initial value for the flush character is taken to be the terminal’s **flush** character.

- erase** If **telnet** is in **localchars** mode (see **toggle localchars** below), and if **telnet** is operating in “character at a time” mode, then when this character is typed, a TELNET EC sequence (see **send ec** above) is sent to the remote system. The initial value for the erase character is taken to be the terminal’s **erase** character.

- kill** If **telnet** is in **localchars** mode (see **toggle localchars** below), and if **telnet** is operating in “character at a time” mode, then when this character is typed, a TEL-

NET EL sequence (see **send el** above) is sent to the remote system. The initial value for the kill character is taken to be the terminal's **kill** character.

eof If **telnet** is operating in "line by line" mode, entering this character as the first character on a line will cause this character to be sent to the remote system. The initial value of the **eof** character is taken to be the terminal's **eof** character.

toggle arguments...

Toggle (between **TRUE** and **FALSE**) various flags that control how **telnet** responds to events. More than one argument may be specified. The state of these flags may be interrogated with the **display** command. Valid arguments are:

localchars

If this is **TRUE**, then the **flush**, **interrupt**, **quit**, **erase**, and **kill** characters (see **set** above) are recognized locally, and transformed into (hopefully) appropriate **TELNET** control sequences (respectively **ao**, **ip**, **brk**, **ec**, and **el**; see **send** above). The initial value for this toggle is **TRUE** in "line by line" mode, and **FALSE** in "character at a time" mode.

autoflush

If **autoflush** and **localchars** are both **TRUE**, then when the **ao**, **intr**, or **quit** characters are recognized (and transformed into **TELNET** sequences; see **set** above for details), **telnet** refuses to display any data on the user's terminal until the remote system acknowledges (via a **TELNET Timing Mark** option) that it has processed those **TELNET** sequences. The initial value for this toggle is **TRUE** if the terminal user had not done an "**stty noflsh**", otherwise **FALSE** (see **stty(1)**).

autosynch

If **autosynch** and **localchars** are both **TRUE**, then when either the **intr** or **quit** characters is typed (see **set** above for descriptions of the **intr** and **quit** characters), the resulting **TELNET** sequence sent is followed by the **TELNET SYNCH** sequence. This procedure **should** cause the remote system to begin throwing away all previously typed input until both of the **TELNET** sequences have been read and acted upon. The initial value of this toggle is **FALSE**.

crmod Toggle carriage return mode. When this mode is enabled, most carriage return characters received from the remote host will be mapped into a carriage return

followed by a line feed. This mode does not affect those characters typed by the user, only those received from the remote host. This mode is not very useful unless the remote host only sends carriage return, but never line feed. The initial value for this toggle is FALSE.

debug Toggles socket level debugging (useful only to the super-user). The initial value for this toggle is FALSE.

options Toggles the display of some internal **telnet** protocol processing (having to do with **TELNET** options). The initial value for this toggle is FALSE.

netdata Toggles the display of all network data (in hexadecimal format). The initial value for this toggle is FALSE.

? Displays the legal **toggle** commands.

BUGS

There is no adequate way for dealing with flow control.

On some remote systems, echo has to be turned off manually when in "line by line" mode.

There is enough settable state to justify a **.telnetrc** file.

No capability for a **.telnetrc** file is provided.

In "line by line" mode, the terminal's **eof** character is only recognized (and sent to the remote system) when it is the first character on a line.

SEE ALSO

inet(3N), **hosts(5)**, **stty(1)**

NAME

test – condition command

SYNOPSIS

test *expr*

DESCRIPTION

test evaluates the expression *expr*, and if its value is true then returns zero exit status; otherwise, a nonzero exit status is returned. **test** returns a nonzero exit if there are no arguments.

The following primitives are used to construct *expr*.

-r <i>file</i>	True if the file exists and is readable.
-w <i>file</i>	True if the file exists and is writable.
-f <i>file</i>	True if the file exists and is not a directory.
-d <i>file</i>	True if the file exists and is a directory.
-s <i>file</i>	True if the file exists and has a size greater than zero.
-L <i>file</i>	True if the file exists and is a soft link.
-t [<i>fildes</i>]	True if the open file whose file descriptor number is <i>fildes</i> (1 by default) is associated with a terminal device.
-z <i>s1</i>	True if the length of string <i>s1</i> is zero.
-n <i>s1</i>	True if the length of the string <i>s1</i> is nonzero.
<i>s1</i> = <i>s2</i>	True if the strings <i>s1</i> and <i>s2</i> are equal.
<i>s1</i> != <i>s2</i>	True if the strings <i>s1</i> and <i>s2</i> are not equal.
<i>s1</i>	True if <i>s1</i> is not the null string.
<i>n1</i> -eq <i>n2</i>	True if the integers <i>n1</i> and <i>n2</i> 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 <i>and</i> operator
-o	binary <i>or</i> operator
(<i>expr</i>)	parentheses for grouping.

-a has higher precedence than **-o**. 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(1), find(1)

NAME

tftp – trivial file transfer protocol

SYNOPSIS

tftp [*-glg!|p|r|w*] *localname host foreignname [mode]*

DESCRIPTION

tftp is the front-end to the Trivial File Transfer Protocol. It enables you to copy files among internet hosts without remote user-level access. A minus sign (-) may be substituted for *localname* in which case the standard input (or output) will be used.

tftp requires a switch to dictate the direction of the file transfer. The recognized switches are:

- put** (*-p, -w*) Write the local file (*localname*) onto the foreign host's file system as *foreignname*. Note that *foreignname* must be quoted if it contains shell special characters. (The word **put**, the switch *-p*, and the switch *-w* are all synonymous).
- get** (*-g, -r*) Read the foreign host's file (*foreignname*) into the local file, *localname*. If *localname* already exists, **tftp** will fail with an appropriate error message.
- get!** (*-g!*) Perform a **tftp get**, overwriting the local file (if it exists). Note that the exclamation point following the command indicates that the command will modify a data structure (in this case, it will overwrite an existing file; the syntax is derived from the Yale/T and MIT/Scheme naming conventions). Within a UNIX shell, the exclamation point must be escaped (usually with a backslash) to avoid shell interpretation.

TRANSFER MODES

Mode may be **netascii**, or **image**. **netascii**, the default mode, transfers the file as standard ascii characters. Image mode transfers the file in binary, with no character conversion.

NOTES

The Domain/OS BSD versions of **tftp** and **tftpd(8C)** are adaptations of the MIT Project Athena implementations of the **tftp** protocol. Domain/OS BSD **tftp** will interface with any RFC783 compliant implementation. Note, however, that the 4.3BSD distribution version of **tftp** does not meet these restrictions.

WARNINGS

tftp and **tftpd(8C)** comprise an implementation of the Trivial File Transfer Protocol described in RFC783. They allow you to quickly copy files among hosts on an internet without regard to ownership or access restrictions. Therefore, the desired security of a system should be considered before allowing **tftp** transactions. In an inspired attempt to prevent accidental destruction of important files, **tftp** requires that remote file names be absolute pathnames (that is, beginning with /) containing the string **"/tftp/"**, but not containing the string **"/./"**.

EXAMPLES

Each of the following examples presumes that there is a host on the internet called

carrara, running a **tftp** server.

1. **tftp -p /tftp/foo carrara /tftp/bar**

Copies the local file **/tftp/foo** to **carrara**, and deposits it in **carrara's /tftp** directory under the name **bar**.

2. **tftp get /tftp/new carrara /tftp/bar**

Copies the remote file (on **carrara**) named **/tftp/bar** to the local file named **/tftp/new**.

3. **tftp -g! /tftp/new carrara /tftp/zed image**

Copies the remote binary file (on **carrara**) named **/tftp/zed** to the local file named **/tftp/new**, overwriting the old copy of **/tftp/new**.

SEE ALSO

tftpd(8C)

Configuring and Managing TCP/IP.

NAME

time – time a command

SYNOPSIS

time *command*

DESCRIPTION

time is used to calculate the real, user, and system times elapsed while a command is executing. The specified *command* is executed by the shell. After execution is completed, **time** prints the following values:

- real** The time elapsed while executing the command.
- user** The number of CPU seconds spent in the program.
- sys** The number of CPU seconds spent in the operating system.

Times are reported in .1 second units, and are printed on standard error.

WARNINGS

This command is **/bin/time**. **csh**(1) and **ksh**(1) each have built-in time functions, with different output formats.

Because real time is calculated in .1 second increments and CPU time is calculated in .004 second increments, the sum of the CPU times may be up to .1 second larger than the elapsed time.

NAME

tip, **cu** – connect to a remote system

SYNOPSIS

```
tip [ -v ] [ -speed ] system-name
tip [ -v ] [ -speed ] phone-number
cu phone-number [ -t ] [ -s speed ] [ -a acu ] [ -l line ] [ -# ]
```

DESCRIPTION

tip and **cu** establish a full-duplex connection to another machine, giving the appearance of being logged in directly on the remote CPU. Naturally, you must have a login (or equivalent) on the machine to which you wish to connect. The preferred interface is **tip**. The **cu** interface is included for those people attached to the “call UNIX” command. This manual page describes only **tip**.

Typed characters are normally transmitted directly to the remote machine (which does the echoing as well). A tilde (~) appearing as the first character of a line is an escape signal; valid commands are listed below.

tip uses the file */etc/remote* to find how to reach a particular system and to find out how it should operate while talking to the system; refer to **remote(5)** for a full description.

When **tip** establishes a connection, it sends out a connection message to the remote system; the default value, if any, is defined in */etc/remote*.

When **tip** prompts for an argument (e.g., during setup of a file transfer) the line typed may be edited with the standard erase and kill characters. A null line (or an interrupt) in response to a prompt aborts the dialogue and returns you to the remote machine.

tip guards against multiple users connecting to a remote system by opening modems and terminal lines with exclusive access, and by honoring the locking protocol used by **uucp(1C)**.

During file transfers **tip** provides a running count of the number of lines transferred. When you use the ~> and ~< commands, the eofread and eofwrite variables are used to recognize end-of-file when reading, and specify end-of-file when writing (see below). File transfers normally depend on tandem mode for flow control. If the remote system does not support tandem mode, set the echocheck variable to indicate that **tip** should synchronize with the remote system on the echo of each transmitted character.

When **tip** must dial a phone number to connect to a system, it will print various messages indicating its actions. **tip** supports the Ventel 212+, Racal-Vadic 3451, and Bizcomp 1031 and 1032 integral call unit/modems.

COMMANDS

```
~CTRL/D    Drop the connection and exit. You may still be logged in on the
             remote machine. The ~. command has the same effect.

~c [name]   Change directory to name. No argument implies your home directory.

~!          Escape to a shell; exiting the shell will return you to tip.
```

- `~>` Copy file from local to remote. **tip** prompts for the name of a local file to transmit.
- `~<` Copy file from remote to local. **tip** prompts first for the name of the file to be sent, then for a command to be executed on the remote machine.
- `~p from [ to ]` Send a file to a remote UNIX host. The put command causes the remote UNIX system to run the command string
`cat > 'to'`
while **tip** sends it the *from* file. If the *to* filename isn't specified, the *from* filename is used. This command is actually a UNIX-specific version of the `~>` command.
- `~t from [ to ]` Take a file from a remote UNIX host. As in the put command, the *to* file defaults to the *from* filename if not specified. The remote host executes the command string
`cat 'from';echo ^A`
to send the file to **tip**.
- `~|` Pipe the output from a remote command to a local UNIX process. The command string sent to the local UNIX system is processed by the shell.
- `~$` Pipe the output from a local UNIX process to the remote host. The command string sent to the local UNIX system is processed by the shell.
- `~#` Send a BREAK to the remote system. For systems that don't support the necessary `ioctl` call, the BREAK is simulated by a sequence of line speed changes and DEL characters.
- `~s` Set a variable (see the discussion below).
- `~CTRL/Z` Stop **tip** (only available with job control).
- `~CTRL/Y` Stop only the local side of **tip** (only available with job control); the remote side of **tip**, the side that displays output from the remote host, is left running.
- `~?` Get a summary of the tilde escapes.

VARIABLES

tip maintains a set of variables which control its operation. Some of these variable are read-only to normal users. (Root is allowed to change anything of interest). Variables may be displayed and set through the `~s` escape. The syntax for variables is patterned after `vi(1)` and `Mail(1)`. Supplying the word **all** as an argument to the `~s` command displays all variables readable by the user.

Variables are numeric (num), string (str), character (char), or Boolean (bool) values. Boolean variables are set merely by specifying their name; they may be reset by prepending an exclamation mark (!) to the name. To set other variable types, concatenate an equal sign (=) and the desired value. The entire assignment must not have any blanks in it.

A single set command may be used to interrogate as well as set a number of variables. You may initialize variables at run time by placing set commands (without the `~s` prefix) in the file `.tiprc` in your home directory.

The following is a list of common variables, their abbreviations, and their default values.

beautify	(bool) Discard unprintable characters when a session is being scripted; abbreviated be .
baudrate	(num) The baud rate at which the connection was established; abbreviated ba .
dialtimeout	(num) When dialing a phone number, the time (in seconds) to wait for a connection to be established; abbreviated dial .
echocheck	(bool) Synchronize with the remote host during file transfer by waiting for the echo of the last character transmitted; default is off.
eofread	(str) The set of characters which signify end-of-transmission during a <code>~<</code> file transfer command; abbreviated eofr .
eofwrite	(str) The string sent to indicate end-of-transmission during a <code>~></code> file transfer command; abbreviated eofw .
eol	(str) The set of characters that indicate an end-of-line. tip will recognize escape characters only after an end-of-line.
escape	(char) The command prefix (escape) character; abbreviated es ; default value is the tilde (<code>~</code>).
exceptions	(str) The set of characters which should not be discarded by the beautification switch; abbreviated ex ; default value is <code>"^\\n\\b"</code> .
force	(char) The character used to force literal data transmission; abbreviated fo ; default value is CTRL/P.
framesize	(num) The amount of data (in bytes) to buffer between file system writes when receiving files; abbreviated fr .
host	(str) The name of the host to which you are connected; abbreviated ho .
prompt	(char) The character which indicates end-of-line on the remote host; abbreviated pr ; default value is <code>"^n"</code> . This value is used to synchronize during data transfers. The count of lines transferred during a file transfer command is based on receipt of this character.
raise	(bool) Upper case mapping mode; abbreviated ra ; default value is off. When this mode is enabled, all lowercase letters will be mapped to

uppercase by **tip** for transmission to the remote machine.

- raisechar** (char) The input character used to toggle uppercase mapping mode; abbreviated **rc**; default value is CTRL/A.
- record** (str) The name of the file in which a session script is recorded; abbreviated **rec**; default value is **tip.record**.
- script** (bool) Session scripting mode; abbreviated **sc**; default is off. When **script** is **true**, **tip** will record everything transmitted by the remote machine in the script record file specified in **record**. If the **beautify** switch is on, only printable ASCII characters will be included in the script file (those characters between 040 and 0177). The variable **exceptions** is used to indicate characters which are an exception to the normal beautification rules.
- tabexpand** (bool) Expand tabs to spaces during file transfers; abbreviated **tab**; default value is false. Each tab is expanded to 8 spaces.
- verbose** (bool) Verbose mode; abbreviated **verb**; default is **true**. When verbose mode is enabled, **tip** prints messages while dialing, shows the current number of lines transferred during a file transfer operations, and more.
- SHELL** (str) The name of the shell to use for the **! command**; the default value is taken from the environment, or is **/bin/sh**.
- HOME** (str) The home directory to use for the **~c command**; the default value is taken from the environment.

OPTIONS

- v** Use verbose mode.
- speed** Specify *speed* as the baud rate for transmission. Overrides the system default.

FILES

- /etc/remote** global system descriptions
- /etc/phones** global phone number data base
- \$(REMOTE)** private system descriptions
- \$(PHONES)** private phone numbers
- ~/tiprc** initialization file.
- /usr/spool/uucp/LCK.*** lock file to avoid conflicts with **uucp**

NOTES

When you are dialing out using **/etc/remote**, you should use one of the **/dev/sio** devices rather than one of the **/dev/tty** devices, because the sio devices do not wait for the carrier. Note, however, the the standard LCK mechanism won't work if one user is using a **/dev/tty** device and another is using a **/dev/sio** device.

SEE ALSO

remote(5), **phones(5)**

NAME

touch – update date last modified of a file

SYNOPSIS

touch [**-c**] [**-f**] *file* ...

DESCRIPTION

touch attempts to set the modified date of each *file*. If *file* exists, **touch** reads a character from the file and writes it back. If *file* does not exist, **touch** will attempt to create it.

OPTIONS

- c** Do not create *file* if it does not exist.
- f** Attempt to force the **touch** in spite of read and write permissions on *file*.

SEE ALSO

utimes(2)

NAME

tpm – set/display touchpad and mouse characteristics

SYNOPSIS

tpm [*options*]

DESCRIPTION

tpm allows you to define characteristics for the touchpad and mouse. The touchpad operates in one of three modes: absolute, relative, and absolute/relative. The mode of operation establishes how movements of your finger on the touchpad affect the position of the cursor on the screen. The three modes differ primarily in how the cursor moves when you lift your finger from the touchpad and then replace it. The mouse operates in relative mode only, and **-s** is the only valid option.

The subsections below describe the three operational modes, as well as the other options.

OPTIONS

If no options are specified, **tpm** displays the current touchpad characteristics.

- a** (default) sSelect absolute mode.
- r** Selects relative mode.
- ar** Selects absolute/relative mode.
- rerange** Sets prescaling factors for touchpad data.
- s x y** Sets scaling factors for *x* and *y*. Values can range from -32768 to 32767. The default scaling factors are 799 for *x* and 1023 for *y* (portrait displays); and 1023 for *x* and 799 for *y* (landscape displays).
- o x y** Sets *x* and *y* as the origin for absolute mode. Values must be in raster units, and can range from 0 to 1023. The default origin is 0,0.
- h n** Sets the hysteresis box size. The value must be in raster units, and can range from 0 to 1023. The default is 5.

DESCRIPTION

Absolute Mode

In absolute mode, using the default scale and origin, the touchpad approximates the screen, so that the top left edge of the touchpad represents cursor positions at the top left edge of the screen. Absolute mode is the default setting. When you place your finger on the touchpad, the cursor jumps to a corresponding position on the screen. Moving your finger across the touchpad moves the cursor across the screen in the same direction.

For example, moving your finger from the top of the touchpad to the bottom moves the cursor from top to bottom on the screen. If you lift your finger from the touchpad, and later touch the pad again, the cursor jumps to a new position on the screen corresponding to the new finger position.

Relative Mode

In relative mode, cursor movements correspond only to finger movements across the touchpad. The cursor does not move when you first place your finger on the touchpad. This differs from absolute mode, where the cursor jumps to a new position when you lift your finger and then replace it. In effect, relative mode causes the touchpad to correspond to different areas of the screen, relative to the current cursor position.

This is the only meaningful mode for a mouse: all movement begins from the current cursor position.

Relative mode is typically used with scale factors less than the defaults. Smaller scale factors mean that the touchpad maps to a smaller area of the screen. For example, scale factors of 200 by 256 specify one-sixteenth of a portrait display's screen area. With small scale factors, relative mode allows fine resolution of the cursor position within a small area.

To reach distant areas on the screen, you can use several strokes on the touchpad or mouse, each stroke moving the cursor closer to its final destination. To assist you in making large movements in relative mode without having to use too many strokes, the speed of cursor movement is artificially accelerated in relation to the speed of finger or mouse movement. Thus, a quick motion moves the cursor farther than a slow, deliberate motion which covers the same distance.

Absolute/Relative Mode

Absolute/relative mode is a combination of absolute and relative modes. It has no meaning for the mouse. In this mode, the first position of your finger on the touchpad establishes the first position of the cursor, as in absolute mode. Moving your finger across the touchpad moves the cursor across the screen. As in relative mode, the scale is typically smaller than the whole screen.

Unlike absolute and relative modes, however, the effect of lifting your finger from the touchpad depends on how long you break contact. If you lift and replace your finger quickly — within a half second — the cursor does not move, and the effect is the same as relative mode. If you break contact for more than a half second, however, the cursor jumps to a new absolute position when you put your finger on the touchpad again.

Absolute/relative mode is useful for jumping the cursor from one place to another, then carefully positioning it in the new area. For example, this mode is commonly used to move the cursor in a jump from one window to another, and then point to a character in the second window.

Prescaling the Touchpad

Raw touchpad data varies slightly from one touchpad to another. Prescaling is, in essence, calibration of the touchpad. Every time you start the node, the touchpad manager prescales the data to determine an exact range for the device.

To prescale, the touchpad manager observes the first thousand points of touchpad data (about 30 seconds of use). During this time, you should try to touch all four edges of the touchpad to ensure that the observed data constitutes an accurate sample. Based on the observed data, the touchpad manager computes a prescaling factor which, when applied to the data, brings all points into the range from -0.05 to 1.05 . This range corresponds to the edges of the screen, plus an overlap of 5%, when multiplied by the default scaling factors. Because of the overlap, you need not touch the internal frame (under the conductive material) to move the cursor to the edge of the screen.

The **-rerange** option invokes prescaling. This option is useful if the first 30 seconds of use did not include the entire range of the touchpad. It is also handy if you change keyboards on a node, and therefore need to reset the prescaling factors without restarting the node.

Scale Factors

The touchpad manager translates, or scales, the data into raster units, which the Display Manager understands. Scale factors, specified with the **-s** option, are applied to the prescaled touchpad data to translate it to raster units for the Display Manager.

The scale factors are multiplied by the prescaled data. The default scale factors are 800 for x and 1024 for y (portrait displays); and 1024 for x and 800 for y (landscape displays). Applying these factors to prescaled data results in numbers from approximately 0 to 799 (for x) and 0 to 1023 (for y) for portrait displays, and vice versa for landscape displays. (Note that the prescaled data allows a 5% overlap, as mentioned in the preceding subsection.)

The default scale factors provide for touchpad data corresponding to the whole screen. Smaller scale factors reduce the area to which the touchpad maps, thereby allowing you to more finely tune the cursor position. This also applies to mouse movement, allowing changes in the apparent motion sensitivity of the device.

Setting the Origin

The origin is the point denoted by the upper left corner of the touchpad, in absolute and absolute/relative mode. In relative mode, the origin has no meaning. By default, the touchpad origin corresponds to the upper left corner of the screen, that is, the point 0,0 in raster units. By changing the origin, you can use the touchpad (in absolute mode) to correspond to a portion of the screen.

This feature is useful for applications that need to move the cursor within a fixed window, rather than across the whole screen. For example, a program that displays a menu in one window might set the origin to the upper left corner of the menu window. Consequently, the touchpad maps onto the menu window instead of the entire screen.

Hysteresis

The hysteresis value defines the dimensions of a box around your finger position on the touchpad or the current position of the mouse. Movement within the box does not change the position of the cursor on the screen.

Specify the hysteresis value in raster units. The touchpad manager compares the value to the difference between the current and previous finger positions on the touchpad or the current and previous positions of the mouse. If the difference is less than the hysteresis value, the cursor does not move. If the difference is greater than the hysteresis value, the hysteresis value is subtracted from the difference and the cursor moves the resulting distance. The default hysteresis value is five.

EXAMPLES

Display current characteristics.

\$ tpm

```
Mode: absolute
Xscale: 1024, Yscale: 800
Hysteresis: 5
Origin: 0, 0
```

Set characteristics to absolute/relative mode with half the default scaling sensitivity (portrait display).

```
$ tpm -ar -s 400 512
```

NAME

tr – translate characters

SYNOPSIS

tr [**-cds**] [*string1* [*string2*]]

DESCRIPTION

tr copies the standard input to the standard output with substitution or deletion of selected characters. Input characters found in *string1* are mapped into the corresponding characters of *string2*. When *string2* is short, it is padded to the length of *string1* by duplicating its last character.

In either string, the notation *a-b* means a range of characters from *a* to *b* in increasing ASCII order. A backslash (\) followed by 1, 2, or 3 octal digits stands for the character whose ASCII code is given by those digits. A backslash followed by any other character stands for that character.

OPTIONS

- c** Complement the set of characters in *string1* with respect to the universe of characters whose ASCII codes are 01 through 0377 octal.
- d** Delete all input characters in *string1*.
- s** Squeeze all strings of repeated output characters in *string2* to single characters.

EXAMPLE

The example below creates a list of all the words in *file1*, one per line in *file2*, where a word is taken to be a maximal string of alphabetic characters. The second string is quoted to protect the backslash from the shell. The 012 is the ASCII code for newline.

```
tr -cs A-Za-z '\012' <file1 >file2
```

BUGS

tr cannot handle ASCII NUL in *string1* or *string2*. It always deletes the NUL from input.

SEE ALSO

ed(1), **expand(1)**, **sed(1)**, **ascii(7)**

NAME

troff – text formatting and typesetting

SYNOPSIS

troff [*options*] ... [*file(s)*] ...

DESCRIPTION

troff formats text in the named files for printing on a phototypesetter; **nroff**(1) is used for typewriter-like devices. The *UNIX Text Processing* manual offers a complete description of the use of each.

If no *file* argument is present, or if you specify a single dash (-), **nroff** reads the standard input.

OPTIONS

The options may appear in any order so long as they appear before the files.

- olist** Print only pages whose page numbers appear in the comma-separated *list* of numbers and ranges. A range *N-M* means pages *N* through *M*; an initial *-N* means from the beginning to page *N*; and a final *N-* means from *N* to the end.
- nN** Number first generated page *N*.
- sN** Stop every *N* pages. **troff** will stop the phototypesetter every *N* pages, produce a trailer to allow changing cassettes, and resume when the typesetter's start button is pressed.
- mname** Prefix the macro file */usr/lib/tmac/tmac.name* to the input *files*.
- raN** Set register *a* (one-character) to *N*.
- i** Read standard input after the input files are exhausted.
- q** Invoke the simultaneous input-output mode of the **rd** request.
- t** Direct output to the standard output instead of the phototypesetter.
- f** Refrain from feeding out paper and stopping phototypesetter at the end of the run.
- w** Wait until phototypesetter is available, if currently busy.
- b** Report whether the phototypesetter is busy or available. No text processing is done.
- a** Send a printable ASCII approximation of the results to the standard output.
- pN** Print all characters in point size *N* while retaining all prescribed spacings and motions, to reduce phototypesetter elapsed time.
- Ffontdir** The directory *fontdir* contains the font width tables instead of the default directory */usr/lib/fonts*. Use this option to produce output for devices besides the phototypesetter.

FILES

`/tmp/ta*` temporary file
`/usr/lib/tmac/tmac.*` standard macro files
`/usr/lib/font/*` font width tables for **troff**

SEE ALSO

`nroff(1)`, `eqn(1)`, `tbl(1)`, `ms(7)`, `me(7)`, `man(7)`, `col(1)`, `colcrt(1)`;
UNIX Text Processing;
J. F. Ossanna, *Nroff/Troff User's Manual*
B. W. Kernighan, *A Troff Tutorial*.

NAME

trty – test a type trait.

SYNOPSIS

trty [*options*] *type_name* *trait_name*

DESCRIPTION

trty tests whether some type supports a given trait. It prints a message and returns a status that a shell can test.

Some built-in types support some traits, other traits have their type managers loaded dynamically. Note that **trty** can test whether a trait has built-in support only on your own node. The **-n** option does not let you test for built-in support in another node. **trty** will test for dynamic support on your own or on another node.

type_name (required) Specify the name of type to be tested.

trait_name (required) Specify the name of the trait to be tested.

OPTION

-n *node_spec* Specify the node on which the type is to be tested. You may also specify the entry directory of a volume mounted for software installation, as shown in the example below. If this option is omitted, the type is tested on the current node.

EXAMPLES

Please note that these examples may not work exactly as shown unless you have these named traits and type installed on your node.

```
$ trty net_iic io_oc
Type "net_iic" does not support trait "io_oc".
```

The following example demonstrates a test on a volume mounted for software installation

```
$ trty net_iic net_hdwr -n/mounted_vol
Type "net_iic" supports trait "net_hdwr".
```

The **trty** program produces a status that a shell (in this case, the Aegis shell) can test for success or failure.

```
$ if trty net_iic net_hdwr >/dev/nul
$_ then args OK
$_ else args Oops
$_ endif
OK
```

```
$ if trty net_iic io_oc >/dev/nul
$_ then args OK
$_ else args Oops
$_ endif
Oops
```

SEE ALSO

dltty(1), inty(1), lty(1)

NAME

true, **false** – provide truth values

SYNOPSIS

true

false

DESCRIPTION

true and **false** are usually used in a Bourne shell script to test for the appropriate status (true or false) before running (or not running) a list of commands.

EXAMPLE

```
while true
do
    command list
done
```

DIAGNOSTICS

true has exit status zero.

SEE ALSO

csh(1), sh(1)

NAME

tr_font – transliterate characters within a font

SYNOPSIS

tr_font *font_name hex_conversion_table*

DESCRIPTION

The **tr_font** command allows you to change the order in which characters appear in fonts. It rearranges the graphic images associated with the characters in *font_name*, according to information in the *hex_conversion_table*. Use it if you create a new font file from two font files that have different character orders.

tr_font only works on fonts already formatted for SR10. It works directly on the font, without creating a backup.

The format for the *hex_conversion_table* file is:

```
src_ordinal dest_ordinal comment
src_ordinal dest_ordinal comment
src_ordinal dest_ordinal comment
```

where *src_ordinal* is the hexadecimal ordinal value of the character whose graphic image is to be used as the source, *dest_ordinal* is the ordinal value of the character which gets the transliterated image, and comment is an optional remark (for the ASCII character set, the hexadecimal ordinal value 41 represents the character 'A'). If the font was created by concatenating two fonts with **cvt_font**, then the hexadecimal ordinal value of the lowest possible character in the second font is 80.

EXAMPLE

The following example rearranges the characters in the SR10 format font file named **courier** according to the information in the *hex_conversion_table theirs_to_ours*.

```
$ tr_font courier theirs_to_ours
```

This is a sample of a *hex_conversion_table* file.

```
A1 A1 !down
A2 A2 cent
A3 A3 sterling
A5 A5 yen
A7 A7 section
A8 A4 currency
AB AB guillemotleft
B6 B6 paragraph
B7 B7 bullet
B8 B8 quitesinglebase
BB BB guill
```

SEE ALSO

cvt_font(1)

NAME

ts – display the module name and time stamp

SYNOPSIS

ts [-nhd] *object_module_name*

DESCRIPTION

ts displays the time stamp and module name stored in an object module. Shown is the time and date that the module was created by one of the linkers or compilers. The time stamp is not affected by copying an object module, so it is a reliable indicator of whether particular object modules are copies of one another.

OPTIONS

-nhd Option does not print the table header. **ts** outputs in tabular format with table header by default.

NAME

tset – terminal dependent initialization

SYNOPSIS

tset [*options*] [**-m** [*ident*][*test baudrate*]:*type*] ... [*type*]

reset [*options*] [**-m** [*ident*][*test baudrate*]:*type*] ... [*type*]

DESCRIPTION

tset sets up your terminal when you first log in to a UNIX system. It does terminal dependent processing such as setting erase and kill characters, setting or resetting delays, sending any sequences needed to properly initialize the terminal, and the like. It first determines the *type* of terminal involved, and then does necessary initializations and mode settings. The type of terminal attached to each UNIX port is specified in the */etc/ttys*(5) database. Type names for terminals may be found in the *termcap*(5) database. If a port is not wired permanently to a specific terminal (not hardwired) it will be given an appropriate generic identifier such as **dialup**.

In the case where no arguments are specified, **tset** simply reads the terminal type out of the environment variable **TERM** and re-initializes the terminal. The rest of this text concerns itself with mode and environment initialization, typically done once at login, and options used at initialization time to determine the terminal type and set up terminal modes.

When used in a start-up script (**.profile** for **sh**(1) users or **.login** for **csh**(1) users) it is desirable to give information about the type of terminal you will usually use on ports that are not hardwired. These ports are identified in */etc/ttys* as **dialup** or **plugboard** or **arpanet**, etc. To specify what terminal type you usually use on these ports, the **-m** (map) option flag is followed by the appropriate port type identifier, an optional baud rate specification, and the terminal type. (The effect is to “map” from some conditions to a terminal type, that is, to tell **tset** “If I’m on this kind of port, guess that I’m on that kind of terminal”.) If more than one mapping is specified, the first applicable mapping prevails. A missing port type identifier matches all identifiers. Any of the alternate generic names given in */etc/termcap* may be used for the identifier.

A *baudrate* is specified as with **stty**(1), and is compared with the speed of the diagnostic output (which should be the control terminal). The baud rate *test* may be any combination of: **>**, **@**, **<**, and **!**; **@** means “at” and **!** inverts the sense of the test. To avoid problems with metacharacters, it is best to place the entire argument to **-m** within ‘ characters; **csh**(1) users must also put a **** before any **!** used here.

Thus

```
tset -m 'dialup>300:adm3a' -m dialup:dw2 -m 'plugboard:?adm3a'
```

causes the terminal type to be set to an **adm3a** if the port in use is a dialup at a speed greater than 300 baud; to a **dw2** if the port is (otherwise) a dialup (that is, at 300 baud or less). If the *type* finally determined by **tset** begins with a question mark, the user is asked if s/he really wants that type. A null response means to use that type; otherwise, another type can be entered which will be used instead. Thus, in the above

case, the user will be queried on a plugboard port as to whether they are actually using an **adm3a**.

If no mapping applies and a final *type* option, not preceded by a **-m**, is given on the command line then that type is used; otherwise the type found in the **/etc/ttys** database will be taken to be the terminal type. This should always be the case for hardwired ports.

It is usually desirable to return the terminal type, as finally determined by **tset**, and information about the terminal's capabilities to a shell's environment. This can be done using the **-o** option; using the Bourne shell, **sh(1)**:

```
export TERM; TERM=`tset -o options...`
```

or using the C shell, **csh(1)**:

```
setenv TERM `tset -o options...`
```

With **csh** it is preferable to use the following command in your **.login** file to initialize the **TERM** and **TERMCAP** environment variables at the same time. You must **set noglob** before invoking the **tset -s** command, and **unset noglob** after the **tset -s** command, as shown in the following example.

```
set noglob
eval `tset -s options...`
unset noglob
```

It is also convenient to make an alias in your **.cshrc**:

```
alias tset `eval `tset -s `!`*``
```

This allows the command:

```
tset 2621
```

to be invoked at any time to set the terminal and environment. Note to Bourne shell users: It is not possible to get this aliasing effect with a shell script, because shell scripts cannot set the environment of their parent. (If a process could set its parent's environment, none of this nonsense would be necessary in the first place.)

These commands cause **tset** to place the name of your terminal in the variable **TERM** in the environment; see **environ(7)**.

Once the terminal type is known, **tset** engages in terminal driver mode setting. This normally involves sending an initialization sequence to the terminal, setting the single character erase (and optionally the line-kill (full line erase)) characters, and setting special character delays. Tab and newline expansion are turned off during transmission of the terminal initialization sequence.

On terminals that can backspace but not overstrike (such as a CRT), and when the erase character is the default erase character ('#' on standard systems), the erase character is changed to **BACKSPACE** (CTRL/H).

OPTIONS

- ec** Set the erase character to be the named character *c* on all terminals, the default being the backspace character on the terminal, usually **^H** (CTRL/H). The character *c* can either be typed directly, or entered using the hat notation used here.
- kc** This option is similar to **-e** but it applies to the line kill character rather than the erase character; *c* defaults to **^X** (for purely historical reasons). The kill character is left alone if **-k** is not specified. The hat notation can also be used for this option.
- ic** This option is similar to **-e** but it applies to the interrupt character rather than the erase character; *c* defaults to **^C** (CTRL/C). The hat notation can also be used for this option.
- The name of the terminal finally decided upon is output on the standard output. This is intended to be captured by the shell and placed in the environment variable **TERM**.
- s** Print the sequence of **cs**h commands to initialize the environment variables **TERM** and **TERMCAP** based on the name of the terminal finally decided upon.
- n** On systems with the Berkeley 4BSD tty driver, specifies that the new tty driver modes should be initialized for this terminal. For a CRT, the **CRTERASE** and **CRTKILL** modes are set only if the baud rate is 1200 or greater. See **tty(4)** for more detail.
- I** Suppresses transmitting terminal initialization strings.
- Q** Suppresses printing the "Erase set to" and "Kill set to" messages.

If **tset** is invoked as **reset**, it will set cooked and echo modes, turn off cbreak and raw modes, turn on newline translation, and restore special characters to a sensible state before any terminal-dependent processing is done. Any special character that is found to be NULL or **^-1** is reset to its default value. All arguments to **tset** may be used with **reset**.

This is most useful after a program dies leaving a terminal in a funny state. You may have to type **"<LF>reset<LF>"** to get it to work since **<CR>** may not work in this state. Often none of this will echo.

EXAMPLES

These examples all assume the Bourne shell and use the **-** option. If you use **cs**h, use one of the variations described above. Note that a typical use of **tset** in a **.profile** or **.login** will also use the **-e** and **-k** options, and often the **-n** or **-Q** options as well. These options have not been included here to keep the examples small. (NOTE: some of the examples given here appear to take up more than one line, for text processing reasons. When you type in real **tset** commands, you must enter them entirely on one line.)

At the moment, you are on a 2621. This is suitable for typing by hand but not for a .profile, unless you are always on a 2621.

```
export TERM; TERM=`tset - 2621`
```

You have an h19 at home which you dial up on, but your office terminal is hardwired and known in /etc/ttys.

```
export TERM; TERM=`tset --m dialup:h19`
```

You have a switch which connects everything to everything, making it nearly impossible to key on what port you are coming in on. You use a vt100 in your office at 9600 baud, and dial up to switch ports at 1200 baud from home on a 2621. Sometimes you use someone else's terminal at work, so you want it to ask you to make sure what terminal type you have at high speeds, but at 1200 baud you are always on a 2621. Note the placement of the question mark, and the quotes to protect the greater than and question mark from interpretation by the shell.

```
export TERM; TERM=`tset --m 'switch>1200:?vt100' -m 'switch<=1200:2621`
```

All of the above entries will fall back on the terminal type specified in /etc/ttys if none of the conditions hold. The following entry is appropriate if you always dial up, always at the same baud rate, on many different kinds of terminals. Your most common terminal is an adm3a. It always asks you what kind of terminal you are on, defaulting to adm3a.

```
export TERM; TERM=`tset - ?adm3a`
```

If the file /etc/ttys is not properly installed and you want to key entirely on the baud rate, the following can be used:

```
export TERM; TERM=`tset --m '>1200:vt100' 2621`
```

Here is a fancy example to illustrate the power of tset and to hopelessly confuse anyone who has made it this far. You dial up at 1200 baud or less on a concept100, sometimes over switch ports and sometimes over regular dialups. You use various terminals at speeds higher than 1200 over switch ports, most often the terminal in your office, which is a vt100. However, sometimes you log in from the university you used to go to, over the ARPANET; in this case you are on an ALTO emulating a dm2500. You also often log in on various hardwired ports, such as the console, all of which are properly entered in /etc/ttys. You want your erase character set to control H, your kill character set to control U, and don't want tset to print the "Erase set to Backspace, Kill set to Control U" message.

```
export TERM; TERM=`tset -e -k`U -Q - -m
'switch<=1200:concept100' -m 'switch:?vt100' -m dialup:concept100
-m arpanet:dm2500`
```

FILES

/etc/ttys

/etc/termcap

port name to terminal type mapping database
terminal capability database

BUGS

The **tset** command is one of the first commands a user must master when getting started on a UNIX system. Unfortunately, it is one of the most complex, largely because of the extra effort the user must go through to get the environment of the log-in shell set. Something needs to be done to make all this simpler, either the **login(1)** program should do this stuff, or a default shell alias should be made, or a way to set the environment of the parent should exist.

This program can't intuit personal choices for erase, interrupt and line kill characters, so it leaves these set to the local system standards.

tset cannot modify Display Manager key definitions for erase, kill, etc.

SEE ALSO

csh(1), **sh(1)**, **stty(1)**, **ttys(5)**, **termcap(5)**, **environ(7)**

NAME

tsort – topological sort

SYNOPSIS

tsort [*file*]

DESCRIPTION

tsort produces, on the standard output, a totally ordered list of items consistent with a partial ordering of items mentioned in the input *file*. If you do not specify a *file*, **tsort** uses the standard input.

The input consists of pairs of items (non-empty strings) separated by blanks. Pairs of different items indicate ordering. Pairs of identical items indicate presence, but not ordering.

BUGS

tsort uses a quadratic algorithm. This is not worth fixing for the typical use of ordering a library archive file.

DIAGNOSTICS

Odd data: there is an odd number of fields in the input file.

SEE ALSO

lorder(1)

NAME

tty – get terminal name

SYNOPSIS

tty [**-s**] [**-p**] [**-l**]

DESCRIPTION

tty prints the pathname of the user's terminal.

OPTIONS

- s** Inhibit printing of the terminal pathname. Allow testing of the exit code only.
- p** If terminal pathname is **/dev/display**, resolve into pad name.
- l** Test for the presence of a synchronous line and, if found, print the line number. This test does not affect the exit code.

EXIT CODES

Returns exit value 0 if standard input is a terminal, 1 if it is not.

Returns exit value 2 if invalid options are specified.

DIAGNOSTICS

Displays "not a tty" if standard input is not a terminal and **-s** is not specified.

Displays "not on an active synchronous line" if the **-l** test fails.

NAME

tz – set or display system time zone

SYNOPSIS

tz [*tz_name* | *utc_delta* [*new_tz*]]

DESCRIPTION

tz sets the system time zone to a known time zone or to an offset from Coordinate Universal Time (utc). If no arguments are specified, **tz** displays the current setting.

tz_name (optional) Specify new time zone. The following are valid names:

Name	Time Zone
EDT	Eastern Daylight Time
EST	Eastern Standard Time
CDT	Central Daylight Time
CST	Central Standard Time
MDT	Mountain Daylight Time
MST	Mountain Standard Time
PDT	Pacific Daylight Time
PST	Pacific Standard Time
GMT	Greenwich Mean Time
UTC	Coordinated Universal Time

Default if omitted: use *utc_delta* argument

utc_delta (optional) Specify positive or negative offset from utc. The plus sign is optional for positive offsets. Format for offset is *hh:mm* (for example, -10:00 for ten hours earlier than, west of, Coordinated Universal Time). Only whole or half hour offsets may be specified. Other fractional offsets produce an error message.

Default if omitted: use *tz_name* argument

new_tz (optional) Specify new time zone name to be assigned to the zone indicated by the *utc_delta* argument. Use this argument to create time zones that are not included in the list above.

Default if omitted: no name assigned

EXAMPLES

Display current time zone.

```
$ tz
  Timezone:      EST
  Delta from UTC: -5:00
```

Set time zone to Pacific Daylight Time.

```
$ tz pdt
```

Create (and set) a time zone named GST that is four and a half hours later than (east of) Coordinated Universal Time.

```
$ tz 4:30 gst
```

NAME

ul - do underlining

SYNOPSIS

ul [**-i**] [**-t** *terminal*] [*name* ...]

DESCRIPTION

ul reads the named files (or standard input if none are given) and translates occurrences of underscores to the sequence that indicates underlining for the terminal in use, as specified by the environment variable **TERM**.

The file **/etc/termcap** is read to determine the appropriate sequences for underlining. If the terminal is incapable of underlining, but is capable of a standout mode, then that is used instead. If the terminal can overstrike, or handles underlining automatically, **ul** degenerates to **cat(1)**. If the terminal cannot underline, underlining is ignored.

OPTIONS

- i** Indicate underlining by a separate line containing appropriate dashes “-”; this option is useful when you want to look at the underlining in an **nroff** output stream on a CRT-terminal.
- t** Override the terminal type specified in the environment.

BUGS

nroff usually outputs a series of backspaces and underlines intermixed with the text to indicate underlining. No attempt is made to optimize the backward motion.

SEE ALSO

man(1), **nroff(1)**, **colcrt(1)**, **more(1)**

NAME

unget – undo a previous **get** of an SCCS file

SYNOPSIS

unget [*-rSID*] [*-s*] [*-n*] *files* ...

DESCRIPTION

unget undoes the effect of executing a **get**(1) command with the *-e* option prior to creating the intended new delta.

If you specify a directory as an argument, **unget** behaves as though each file in the directory was specified as a named file, except that it silently ignores non-SCCS and unreadable files.

If you supply a dash (*-*) in place of a file or directory, **unget** reads the standard input, interpreting each line as the name of an SCCS file to be processed.

The options below apply independently to each named file.

OPTIONS

- rSID* Uniquely identify which delta is no longer intended, i.e., specified by **get**(1) as the new delta. This is necessary only if two or more outstanding executions of **get**(1) for editing on the same SCCS file were done under the same log-in name. A diagnostic results if the specified SID (SCCS IDentification number) is ambiguous, or if it is required but was omitted on the command line.
- s* Suppress the printout, on the standard output, of the intended delta's SID.
- n* Retain the file secured through the execution of **get**(1). This file is normally removed from the current directory.

DIAGNOSTICS

Use **help**(1) for explanations.

SEE ALSO

admin(1), **delta**(1), **get**(1), **help**(1), **sact**(1) **scs**(1)

NAME

unifdef – remove `ifdef`'ed lines

SYNOPSIS

unifdef [**-t**] [**-l**] [**-c**] [**-Dsym** | **-Usym** | **-idsym** | **-iusym**] ... [*file*]

DESCRIPTION

unifdef is useful for removing `ifdef`'ed lines from a file while otherwise leaving the file alone. **unifdef** is like a stripped-down C preprocessor: it is smart enough to deal with the nested `ifdefs`, comments, single and double quotes of C syntax so that it can do its job, but it doesn't do any including or interpretation of macros. Neither does it strip out comments, though it recognizes and ignores them.

unifdef copies its output to the standard output and will take its input from the standard input if no *file* argument is given.

OPTIONS

- t** Refrain from attempting to recognize comments and single and double quotes. Used for plain text files (not C code).
- l** Replace removed lines with blank lines instead of deleting them.
- c** Complement the operation of **unifdef**: the lines that would have been removed or blanked are retained and vice versa.
- Dsym** Specify that *sym* is defined.
- Usym** Specify that *sym* is undefined. The lines inside those `ifdefs` will be removed. The `ifdef`, `ifndef`, `else`, and `endif` lines associated with *sym* will also be removed. `ifdefs` involving symbols you don't specify are untouched and copied out along with their associated `ifdef`, `else`, and `endif` lines. If an `ifdef X` occurs nested inside another `ifdef X`, then the inside `ifdef` is treated as if it were an unrecognized symbol. If the same symbol appears in more than one argument, only the first occurrence is significant.
- idsym** Specify that *sym* is defined. If you use `ifdefs` to delimit non-C lines, such as comments or code which is under construction, then you must use this option tell **unifdef** which symbols are used for that purpose so that it won't try to parse for quotes and comments in those `ifdef`'ed lines. The lines inside are ignored but copied out.
- iusym** Specify that *sym* is undefined. This option complements **-id**.

DIAGNOSTICS

Premature EOF, inappropriate else or endif.

Exit status is 0 if output is exact copy of input, 1 if not, 2 if trouble.

BUGS

Does not know how to deal with **cpp** constructs such as

```
#if      defined(X) || defined(Y)
```

SEE ALSO

diff(1)

NAME

uniq – report repeated lines in a file

SYNOPSIS

uniq [**-udc** [**+n**] [**-n**]] [*input* [*output*]]

DESCRIPTION

uniq reads the input file, comparing adjacent lines. Normally it removes the second and succeeding copies of repeated lines, and writes the remainder on the output file. Repeated lines must be adjacent in order to be found; refer to **sort(1)** for more information on this. The normal mode output is the union of the **-u** and **-d** mode outputs.

OPTIONS

- u** Output only the lines not repeated in the original file.
- d** Write one copy of only the repeated lines.
- c** Supersede the **-u** and **-d** options. Generate an output report in default style, but with each line preceded by a count of the number of times it occurred.
- n** Ignore the first *n* fields, together with any blanks before each. A field is defined as a string of nonspace, nontab characters separated from adjacent characters by tabs and spaces.
- +n** Ignore the first *n* characters. Fields are skipped before characters.

SEE ALSO

comm(1), **sort(1)**

NAME

units – conversion program

SYNOPSIS

units

DESCRIPTION

units converts quantities expressed in various standard scales to their equivalents in other scales. It works interactively, prompting you for information.

You specify a quantity as a multiplicative combination of units optionally preceded by a numeric multiplier. Powers are indicated by suffixed positive integers; division is indicated by the usual sign.

units does only multiplicative scale changes; thus, it can convert Kelvin to Rankine, but not Celsius to Fahrenheit. It recognizes most familiar units, abbreviations, and metric prefixes, along with some others that include the following

pi	ratio of circumference to diameter
c	speed of light
e	charge on an electron
g	acceleration of gravity
force	same as g
mole	Avogadro's number
water	pressure head per unit height of water
au	astronomical unit

A **pound** is a unit of mass. Compound names are run together (e.g., **lightyear**). British units that differ from their U.S. counterparts are prefixed with **br** (e.g., **brgallon**). Currency is prefixed with the name of the country it comes from (e.g., **belgiumfranc**, **britainpound**).

For a complete list of units, examine the file **/usr/lib/units**.

EXAMPLES

```
% units
you have: inch
you want: cm
          * 2.540000e+00
          / 3.937008e-01

you have: 15 lbs force/in2
you want: atm
          * 1.020689e+00
          / 9.797299e-01
```

FILES

/usr/lib/units complete list of units

UNITS(1)

BSD

UNITS(1)

BUGS

The `/usr/lib/units` database does not automatically reflect fluctuations in international currency exchanges.

NAME

uptime – show how long system has been up

SYNOPSIS

uptime

DESCRIPTION

uptime prints the current time, the length of time the system has been up, and the average number of jobs in the run queue over the last 1, 5 and 15 minutes.

NAME

users – compact list of users who are on a node

SYNOPSIS

users

DESCRIPTION

users lists the log-in names of the users currently logged into your node in a compact, one-line format.

FILES

/etc/utmp

SEE ALSO

who(1)

NAME

uucp, uulog, uuname – UNIX-to-UNIX system copy

SYNOPSIS

uucp [*options*] *source-files destination-file*

uulog [*options*] *-s system*

uulog [*options*] *system*

uulog [*options*] *-f system*

uuname [*-l*] [*-c*]

DESCRIPTION

uucp copies files named by the *source-file* arguments to the *destination-file* argument. A filename may be a pathname on a machine, or may have the following form:

system name!pathname

where *system name* is taken from a list of system names that **uucp** knows about. The *system name* may also be a list of names such as:

system name!system name!...!system name!pathname

in which case an attempt is made to send the file via the specified route, to the destination. See NOTES and BUGS below for restrictions. Care should be taken to ensure that intermediate nodes in the route are willing to forward information (see NOTES below for restrictions).

The question mark (?), asterisk (*), and bracketed ellipsis ([...]) shell metacharacters appearing in *pathname* are expanded on the appropriate system.

Pathnames may be one of the following (anything else is prefixed by the current directory):

- a full pathname
- a pathname preceded by *~user* where *user* is a log-in name on the specified system and is replaced by that user's log-in directory
- a pathname preceded by *~/destination* where *destination* is appended to */usr/spool/uucppublic*; (NOTE: This destination will be treated as a filename unless more than one file is being transferred by this request or the destination is already a directory. To ensure that it is a directory, follow the destination with a slash mark (/). For example, *~/dan/* as the destination will make the directory */usr/spool/uucppublic/dan* if it does not exist and put the requested file(s) in that directory).

If the result is an erroneous pathname for the remote system, the copy will fail. If the *destination-file* is a directory, the last part of the *source-file* name is used.

uucp preserves execute permissions across the transmission, and gives 0666 read and write permissions. See **chmod(2)** for more information about permissions. All files received by **uucp** will be owned by **uucp**.

uulog queries a summary log of **uucp** or **uuxqt** transactions in the files **/usr/spool/uucp/.Log/uucico/system**, or **/usr/spool/uucp/.Log/uuxqt/system**.

uname lists the names of systems known to **uucp**.

The BSD version of **uucp** supports the Vadic Autodialer.

OPTIONS

The following options are interpreted by **uucp** only:

- c** Don't copy the local file to the spool directory for transfer to the remote machine (default).
- C** Force the copy of local files to the spool directory for transfer.
- d** Make all necessary directories for the file copy (default).
- f** Do not make intermediate directories for the file copy.
- ggrade** *Grade* is a single letter/number; lower ascii sequence characters will cause the job to be transmitted earlier during a particular conversation.
- j** Output the job identification ASCII string on the standard output. This job identification can be used by **uustat** to obtain the status or terminate a job.
- m** Send mail to the requester when the copy is completed. The **-m** option only works sending files or receiving a single file. Receiving multiple files specified by special shell characters **? * [...]** will not activate the **-m** option.
- nuser** Notify *user* on the remote system that a file was sent.
- r** Don't start the file transfer, just queue the job.
- sfile** Report status of the transfer to *file*. Note that the *file* must be a full pathname.
- xdebug_level** Produce debugging output on standard output. The *debug_level* is a number between 0 and 9; higher numbers give more detailed information. (Debugging will not be available if **uucp** was compiled with **-DSMALL**.)

uulog Options

The following options are used by **uulog** *only*; they cause **uulog** to print logging information:

- sys** Print information about file transfer work involving system *sys*.
- fsystem** Do a **tail -f** of the file transfer log for *system*. (You must press BREAK to exit this function.)

Other options used in conjunction with the above **uulog** options:

- x** Look in the *uuxqt* log file for the given system.
- number** Indicates that a **tail** command of *number* lines should be executed.

uname Options

The following options are used by **uname** *only*:

- c** Return the names of systems known to **cu**. This list should be the same as the list of systems known to **uucp**, unless your machine is using different *Systems* files for **cu** and **uucp**. See the *Sysfiles* file.
- l** Return the local system name.

NOTES

The domain of remotely accessible files may (and for obvious security reasons, usually should) be severely restricted. You will very likely not be able to fetch files by pathname. Ask a responsible person on the remote system to send them to you. For the same reasons, you will probably not be able to send files to arbitrary pathnames. As distributed, the remotely accessible files are those whose names begin with */usr/spool/uucppublic* (equivalent to *~*).

The forwarding of files through other systems may not be compatible with the previous version of **uucp**. If forwarding is used, all systems in the route must have the same version of **uucp**.

BUGS

Protected files and files in protected directories that are owned by the requestor can be sent by **uucp**. However, if the requestor is **root**, and the directory is not searchable by "other" or the file is not readable by "other", the request will fail.

FILES

- /usr/spool/uucp* spool directory
- /usr/spool/uucppublic* public directory for receiving and sending (PUB-DIR)
- /usr/lib/uucp/** other data and program files

uucp looks in */etc/net/uname* to determine the site name for the local system. If there is no such file, it assumes **apollo1**.

SEE ALSO

mail(1), *uustat(1C)*, *uux(1C)*, *uuxqt(8)*, *chmod(2)*;
also refer to the discussion of **uucp** in *Managing BSD System Software*.

NAME

uencode, **uudecode** – encode/decode a binary file for transmission via mail

SYNOPSIS

```
uencode [ source ] remotedest | mail sys1!sys2!...!decode  
uudecode [ file ]
```

DESCRIPTION

uencode and **uudecode** are used to send a binary file via **uucp**(1C) or other methods of sending mail. This combination can be used over indirect mail links even when **uusend**(1C) is not available.

uencode takes the named source file (the default is standard input) and produces an encoded version on the standard output. The encoding uses only printing ASCII characters, and includes the mode of the file and the *remotedest* for re-creation on the remote system.

uudecode reads an encoded file, strips off any leading and trailing lines added by mailers, and recreates the original file with the specified mode and name.

The intent is that all mail to the user *decode* should be filtered through the **uudecode** program. This way the file is created automatically without human intervention. This is possible on the uucp network by either using **sendmail**(8) or by making **rmail**(1) be a link to **/usr/ucb/mail** instead of **/bin/mail**. In each case, an alias must be created in a master file to get the automatic invocation of **uudecode**.

If these facilities are not available, the file can be sent to a user on the remote machine who can **uudecode** it manually.

The encode file has an ordinary text form and can be edited by any text editor to change the mode or remote name.

EXAMPLE

The *remotedest* is the pathname of the file to create on the remote system, for example,

```
% uencode /kate/bin/prog1 /tmp/kate.prog1.
```

BUGS

The file is expanded by 35% (three bytes become four plus control information), increasing the transmission time.

The user on the remote system who is invoking **uudecode** (often **uucp**) must have write permission on the specified file.

SEE ALSO

binmail(1), **mail**(1), **uusend**(1C), **uucp**(1C), **uux**(1C), **uencode**(5);
Managing BSD System Software.

NAME

uustat - **uucp** status inquiry and job control

SYNOPSIS

```
uustat [-a]
uustat [-m]
uustat [-p]
uustat [-q]
uustat [ -kjobid ]
uustat [ -rjobid ]
uustat [ -ssystem ] [ -uuser ]
```

DESCRIPTION

uustat will display the status of, or cancel, previously specified **uucp** commands, or provide general status on **uucp** connections to other systems.

OPTIONS

Only one of the following options can be specified with **uustat** per command execution.

- a Output all jobs in the queue.
- m Report the status of accessibility of all machines.
- p Execute a **ps -flp** for all the process-ids that are in the lock files.
- q List the jobs queued for each machine. If a status file exists for the machine, its date, time and status information are reported. In addition, if a number appears in () next to the number of C or X files, it is the age in days of the oldest C/X. file for that system. The "Retry" field represents the number of hours until the next possible call. The "Count" is the number of failure attempts. NOTE: for systems with a moderate number of outstanding jobs, this could take 30 seconds or more of real-time to execute.
- k*jobid* Kill the **uucp** request whose job identification is *jobid*. The killed **uucp** request must belong to the person issuing the **uustat** command unless one is the super-user.
- r*jobid* Rejuvenate *jobid*. The files associated with *jobid* are touched so that their modification time is set to the current time. This prevents the cleanup daemon from deleting the job until the jobs modification time reaches the limit imposed by the daemon.

Either or both of the following options can be specified with **uustat**.

- ssys Report the status of all **uucp** requests for remote system *sys*.
- uuser Report the status of all **uucp** requests issued by *user*.

EXAMPLES

The following example shows the output produced by the **-q** option.

```
eagle      3C      04/07-11:07      NO DEVICES AVAILABLE
mh3bs3     2C      07/07-10:42      SUCCESSFUL
```

The above output tells how many command files are waiting for each system. Each command file may have zero or more files to be sent (zero means to call the system and see if work is to be done). The date and time refer to the previous interaction with the system, and are followed by the status of the interaction.

Output for both the **-s** and **-u** options has the following format:

```
eaglen0000  4/07-11:01:03  (POLL)
eagleN1bd7  4/07-11:07      S      eagle  dan   522 /usr/dan/A
eagleC1bd8  4/07-11:07      S      eagle  dan   59 D.al2ce4924
              4/07-11:07      S      eagle  dan   rmail mike
```

With the **-s** and **-u** options, the first field is the *jobid* of the job. This is followed by the date/time. The next field is either an "S" or "R" depending on whether the job is to send or request a file. This is followed by the user-id of the user who queued the job. The next field contains the size of the file, or in the case of a remote execution (**rmail** – the command used for remote mail), the name of the command. When the size appears in this field, the filename is also given. This can either be the name given by the user or an internal name (e.g., **D.alce4924**) that is created for data files associated with remote executions (**rmail** in this example).

When no options are given, **uustat** outputs the status of all **uucp** requests issued by the current user.

FILES

/usr/spool/uucp/* Spool directories

SEE ALSO

uucp(1C);
Managing BSD System Software.

NAME

uuto, uupick – public UNIX-to-UNIX system file copy

SYNOPSIS

uuto [*options*] *source-files destination*

uupick [*-s system*]

DESCRIPTION

uuto sends *source-files* to *destination*. **uuto** uses the **uucp**(1C) facility to send files, while it allows the local system to control the file access. A *source-file* name is a pathname on your machine. Destination has the form

system!user

where *system* is taken from a list of system names that **uucp** knows about (see **uname**(1)); *user* is the log-in name of someone on the specified system.

The files (or sub-trees if directories are specified) are sent to PUBDIR on *system*, where PUBDIR is a public directory defined in the **uucp** source. By default this directory is */usr/spool/uucppublic*. Specifically the files are sent to

PUBDIR/receive/user/mysystem/files.

The destined recipient is notified by **mail**(1) of the arrival of files.

Two **uuto** options are available:

-p Copy the source file into the spool directory before transmission.

-m Send mail to the sender when the copy is complete.

uupick accepts or rejects the files transmitted to the user. Specifically, **uupick** searches PUBDIR for files destined for the user. For each entry (file or directory) found, the following message is printed on the standard output:

```
from system: [file file-name] [dir dirname] ?
```

uupick then reads a line from the standard input to determine the disposition of the file:

<new-line> Go on to the next entry.

d Delete the entry.

m [*dir*] Move the entry to named directory *dir*. If *dir* is not specified as a complete pathname (in which \$HOME is legitimate), a destination relative to the current directory is assumed. If no destination is given, the default is the current directory.

a [*dir*] The same as **m** except that it moves all the files sent from *system*.

p Print the content of the file.

q Stop.

EOT (CTRL/D) Same as **q**.

!command Escape to the shell to do *command*.

***** Print a command summary.

uupick invoked with the **-ssystem** option will only search the PUBDIR for files sent from *system*.

FILES

PUBDIR /usr/spool/uucppublic Public directory

WARNINGS

To send files that begin with a dot (for example, **.profile**) the files must be qualified with a dot. For example: **.profile**, **.prof***, **.profil?** are correct; whereas ***prof***, **?profile** are incorrect.

SEE ALSO

mail(1), **uucp(1C)**, **uustat(1C)**, **uux(1C)**;
uucleanup(8),
Managing BSD System Software.

NAME

uux – UNIX-to-UNIX system command execution

SYNOPSIS

uux [*options*] *command-string*

DESCRIPTION

uux gathers zero or more files from various systems, executes a command on a specified system, and then sends standard output to a file on a specified system.

The *command-string* is made up of one or more arguments that look like a shell command line, except that the command and filenames may be prefixed by *system-name*!. A null *system-name* is interpreted as the local system.

Filenames may be one of the following: a full path name a path name preceded by *xxx* where *xxx* is a login name on the specified system and is replaced by that user's login directory; anything else is prefixed by the current directory. <PP Any special shell characters such as <>| should be quoted, either by quoting the entire *command-string*, or by quoting the special characters as individual arguments.

uux attempts to get all files to the execution system. For output files, the filename must be escaped using parentheses. **uux** notifies you if the requested command on the remote system was disallowed. This notification can be turned off by the **-n** option. The response comes by remote mail from the remote machine.

OPTIONS

The following options are interpreted by **uux**:

- The standard input to **uux** is made the standard input to the *command-string*.
- aname* Use *name* as the user identification replacing the initiator user-id. (Notification will be returned to the user.)
- b Return whatever standard input was provided to the **uux** command if the exit status is non-zero.
- c Do not copy local file to the spool directory for transfer to the remote machine (default).
- C Force the copy of local files to the spool directory for transfer.
- ggrade* *Grade* is a single letter/number; lower ASCII sequence characters will cause the job to be transmitted earlier during a particular conversation.
- j Output the jobid ASCII string on the standard output which is the job identification. This job identification can be used by **uustat(1)** to obtain the status or terminate a job.
- n Do not notify the user if the command fails.
- p Same as -: The standard input to *uux* is made the standard input to the *command-string*.

-r	Do not start the file transfer, just queue the job.
-sfile	Report status of the transfer in <i>file</i> .
-xdebug_level	Produce debugging output on the standard output. The <i>debug_level</i> is a number between 0 and 9; higher numbers give more detailed information.
-z	Send success notification to the user.

NOTES

For security reasons, most installations limit the list of commands executable on behalf of an incoming request from **uux**, permitting only the receipt of mail (see **mail(1)**). (Remote execution permissions are defined in **/usr/lib/uucp/Permissions**.)

EXAMPLES

The command

```
uux a!cut -f1 b!/usr/file \(!/usr/file)
```

gets **/usr/file** from system **b** and sends it to system **a**, performs a **cut(1)** command on that file and sends the result of the **cut** command to system **c**.

The command

```
uux "!diff usg!/usr/dan/file1 pwba!/a4/dan/file2 > !~/dan/file.diff"
```

will get the **file1** and **file2** files from the machines **usg** and **pwba**, execute a **diff(1)** command, and put the results in **file.diff** in the local **PUBDIR/dan/** directory.

BUGS

Only the first command of a shell pipeline may have a *system-name*!. All other commands are executed on the system of the first command.

The use of the shell metacharacter ***** will probably not do what you want it to do. The shell tokens **<<** and **>>** are not implemented.

The execution of commands on remote systems takes place in an execution directory known to the **uucp** system. All files required for the execution will be put into this directory unless they already reside on that machine. Therefore, the simple filename (without path or machine reference) must be unique within the **uux** request. The following command will NOT work:

```
uux "a!diff b!/usr/dan/xyz c!/usr/dan/xyz > !xyz.diff"
```

but the command

```
uux "a!diff a!/usr/dan/xyz c!/usr/dan/xyz > !xyz.diff"
```

will work. (If **diff(1)** is a permitted command.)

Protected files and files that are in protected directories that are owned by the requestor can be sent in commands using **uux**. However, if the requestor is root, and the directory is not searchable by "other", the request will fail.

FILES

<code>/usr/lib/uucp/spool</code>	Spool directories
<code>/usr/lib/uucp/Permissions</code>	Remote execution permissions
<code>/usr/lib/uucp/*</code>	Other data and programs

SEE ALSO

mail(1), uucp(1C), uustat(1C),
Managing BSD System Software.

NAME

uwm - a window manager for X

SYNTAX

uwm [-display *display*] [-f *filename*]

DESCRIPTION

The *uwm* program is a window manager for X.

When *uwm* is invoked, it searches a predefined search path to locate any *uwm* startup files. If no startup files exist, *uwm* initializes its built-in default file.

If startup files exist in any of the following locations, it adds the variables to the default variables. In the case of contention, the variables in the last file found override previous specifications. Files in the *uwm* search path are:

/usr/lib/X11/uwm/system.uwmrc
\$HOME/.uwmrc

To use only the settings defined in a single startup file, include the variables **resetbindings**, **resetmenus**, **resetvariables** at the top of that specific startup file.

OPTIONS

-f *filename*

Names an alternate file as a *uwm* startup file.

STARTUP FILE VARIABLES

Variables are typically entered first, at the top of the startup file. By convention, **resetbindings**, **resetmenus**, and **resetvariables** head the list.

autoselect/noautoselect

places the menu cursor in the first menu item. If unspecified, the menu cursor is placed in the menu header when the menu is displayed.

background=*color*

specifies the default background color for popup sizing windows, menus, and icons. The default is to use the WhitePixel for the current screen.

bordercolor=*color*

specifies the default border color for popup sizing windows, menus, and icons. The default is to use the BlackPixel for the current screen.

borderwidth=*pixels*

specifies the default width in pixels for borders around popup sizing windows, menus, and icons. The default is 2.

delta=*pixels*

indicates the number of pixels the cursor is moved before the action is interpreted by the window manager as a command. (Also refer to the **delta** mouse action.)

- foreground=***color* specifies the default foreground color for popup sizing windows, menus, and icons. The default is to use the BlackPixel for the current screen.
- freeze/nofreeze** locks all other client applications out of the server during certain window manager tasks, such as move and resize.
- grid/nogrid** displays a finely-ruled grid to help you position an icon or window during resize or move operations.
- hiconpad=***pixels* indicates the number of pixels to pad an icon horizontally. The default is five pixels.
- hmenupad=***pixels* indicates the amount of space in pixels that each menu item is padded to the left and to the right of the text.
- borderwidth=***pixels* indicates the width in pixels of the border surrounding icons.
- iconfont=***fontname* names the font that is displayed within icons. Font names for a given server can be obtained using *xlsfonts(1)*.
- maxcolors=***number* limits the number of colors the window manager can use in a given invocation. If set to zero, or not specified, *uwm* assumes no limit to the number of colors it can take from the color map. **maxcolors** counts colors as they are included in the file.
- mborderwidth=***pixels* indicates the width in pixels of the border surrounding menus.
- normali/nonormali** places icons created with **f.newiconify** within the root window, even if it is placed partially off the screen. With **nonormali** the icon is placed exactly where the cursor leaves it.
- normalw/nonormalw** places window created with **f.newiconify** within the root window, even if it is placed partially off the screen. With **nonormalw** the window is placed exactly where the cursor leaves it.
- push=***number* moves a window *number* pixels or $1/\textit{number}$ times the size of the window, depending on whether **pushabsolute** or **pushrelative** is specified. Use this variable in conjunction with **f.pushup**, **f.pushdown**, **f.pushright**, or **f.pushleft**.
- pushabsolute/pushrelative** **pushabsolute** indicates that the number entered with **push** is equivalent to pixels. When an **f.push** (left, right, up, or down) function is called, the window is moved exactly that number of pixels.

pushrelative indicates that the number entered with the push variable represents a relative number. When an **f.push** function is called, the window is invisibly divided into the number of parts you entered with the push variable, and the window is moved one part.

resetbindings, resetmenus, and resetvariables

resets all previous function bindings, menus, and variable entries, specified in any startup file in the *uwm* search path, including those in the default environment. By convention, these variables are entered first in the startup file.

resizefont=fontname

identifies the font of the indicator that displays dimensions in the corner of the window as you resize windows. See *xlsfonts(1)* for obtaining font names.

resizerelative/noresizerelative

indicates whether or not resize operations should be done relative to moving edge or edges. By default, the dynamic rectangle uses the actual pointer location to define the new size.

reverse/noreverse

defines the display as black characters on a white background for the window manager windows and icons.

viconpad=pixels indicates the number of pixels to pad an icon vertically. Default is five pixels.

vmenupad=pixels

indicates the amount of space in pixels that the menu is padded above and below the text.

volume=number increases or decreases the base level volume set by the *xset(1)* command. Enter an integer from 0 to 7, 7 being the loudest.

zap/nozap causes ghost lines to follow the window or icon from its previous default location to its new location during a move or resize operation.

BINDING SYNTAX

function=[control key(s)]:[context]:mouse events:" menu name "

Function and mouse events are required input. Menu name is required with the *f.menu* function definition only.

Function

f.beep emits a beep from the keyboard. Loudness is determined by the volume variable.

f.circledown causes the top window that is obscuring another window to drop to the bottom of the stack of windows.

f.circleup exposes the lowest window that is obscured by other windows.

f.continue	releases the window server display action after you stop action with the f.pause function.
f.focus	directs all keyboard input to the selected window. To reset the focus to all windows, invoke <i>f.focus</i> from the root window.
f.iconify	when implemented from a window, this function converts the window to its respective icon. When implemented from an icon, <i>f.iconify</i> converts the icon to its respective window.
f.kill	kills the client that created a window.
f.lower	lowers a window that is obstructing a window below it.
f.menu	invokes a menu. Enclose 'menu name' in quotes if it contains blank characters or parentheses. f.menu = [control key(s)] : [context] : mouse events: " menu name "
f.move	moves a window or icon to a new location, which becomes the default location.
f.moveopaque	moves a window or icon to a new screen location. When using this function, the entire window or icon is moved to the new screen location. The grid effect is not used with this function.
f.newiconify	allows you to create a window or icon and then position the window or icon in a new default location on the screen.
f.pause	temporarily stops all display action. To release the screen and immediately update all windows, use the f.continue function.
f.pushdown	moves a window down. The distance of the push is determined by the push variables.
f.pushleft	moves a window to the left. The distance of the push is determined by the push variables.
f.pushright	moves a window to the right. The distance of the push is determined by the push variables.
f.pushup	moves a window up. The distance of the push is determined by the push variables.
f.raise	raises a window that is being obstructed by a window above it.
f.refresh	results in exposure events being sent to the window server clients for all unobscured or partially obscured windows. The windows will not refresh correctly if the exposure events are not handled properly.
f.resize	resizes an existing window. Note that some clients, notably editors, react unpredictably if you resize the window while the client is running.
f.restart	causes the window manager application to restart, retracing the <i>uwm</i> search path and initializing the variables it finds.

Control Keys

By default, the window manager uses meta as its control key. It can also use ctrl, shift, lock, or null (no control key). Control keys must be entered in lower case, and can be abbreviated as: c, l, m, s for ctrl, lock, meta, and shift, respectively.

You can bind one, two, or no control keys to a function. Use the bar (|) character to combine control keys.

Note that client applications other than the window manager use the shift as a control key. If you bind the shift key to a window manager function, you can not use other client applications that require this key.

Context

The context refers to the screen location of the cursor when a command is initiated. When you include a context entry in a binding, the cursor must be in that context or the function will not be activated. The window manager recognizes the following four contexts: icon, window, root, (null).

The root context refers to the root, or background window. A (null) context is indicated when the context field is left blank, and allows a function to be invoked from any screen location. Combine contexts using the bar (|) character.

Mouse Buttons

Any of the following mouse buttons are accepted in lower case and can be abbreviated as l, m, or r, respectively: left, middle, right.

With the specific button, you must identify the action of that button. Mouse actions can be:

- down** function occurs when the specified button is pressed down.
- up** function occurs when the specified button is released.
- delta** indicates that the mouse must be moved the number of pixels specified with the delta variable before the specified function is invoked. The mouse can be moved in any direction to satisfy the delta requirement.

MENU DEFINITION

After binding a set of function keys and a menu name to **f.menu**, you must define the menu to be invoked, using the following syntax:

```
menu = " menu name " {
  "item name" : "action"
  .
  .
  .
}
```

Enter the menu name exactly the way it is entered with the **f.menu** function or the window manager will not recognize the link. If the menu name contains blank strings, tabs or parentheses, it must be quoted here and in the **f.menu** function entry. You can enter as many menu items as your screen is long. You cannot scroll within menus.

Any menu entry that contains quotes, special characters, parentheses, tabs, or strings of blanks must be enclosed in double quotes. Follow the item name by a colon (:).

Menu Action

Window manager functions

Any function previously described. E.g., **f.move** or **f.iconify**.

Shell commands

Begin with an exclamation point (!) and set to run in background. You cannot include a new line character within a shell command.

Text strings

Text strings are placed in the window server's cut buffer.

Strings starting with an up arrow (^) will have a new line character appended to the string after the up arrow (^) has been stripped from it.

Strings starting with a bar character (|) will be copied as is after the bar character (|) has been stripped.

Color Menus

Use the following syntax to add color to menus:

```
menu = "menu name" (color1:color2:color3:color4) {
  "item name" : (color5 :color6) : " action "
  .
  .
  .
}
```

color1 Foreground color of the header.

color2 Background color of the header.

color3 Foreground color of the highlighter, the horizontal band of color that moves with the cursor within the menu.

color4 Background color of the highlighter.

color5 Foreground color for the individual menu item.

color6 Background color for the individual menu item.

Color Defaults

Colors default to the colors of the root window under any of the following conditions:

- 1) If you run out of color map entries, either before or during an invocation of *uwm*.
- 2) If you specify a foreground or background color that does not exist in the RGB color database of the server (see */usr/lib/X11/rgb.txt* for a sample) both the foreground and background colors default to the root window colors.
- 3) If you omit a foreground or background color, both the foreground and background colors default to the root window colors.

4) If the total number of colors specified in the startup file exceeds the number specified in the *maxcolors* variable.

5) If you specify no colors in the startup file.

EXAMPLES

The following sample startup file shows the default window manager options:

```
# Global variables
#
resetbindings; resetvariables; resetmenus
autoselect
delta=25
freeze
grid
hiconpad=5
hmenupad=6
iconfont=oldeng
menufont=timroml2b
resizefont=9x15
viconpad=5
vmenupad=3
volume=7
#
# Mouse button/key maps
#
# FUNCTION      KEYS  CONTEXT  BUTTON  MENU(if any)
# =====      ====  =====  =====  =====
f.menu =        meta  :        :left down  : "WINDOW OPS"
f.menu =        meta  :        :middle down : "EXTENDED WINDOW OPS"
f.move =        meta  :w|i     :right down
f.circleup =    meta  :root   :right down
#
# Menu specifications
#
menu = "WINDOW OPS" {
  "(De)Iconify":    f.iconify
  Move:             f.move
  Resize:           f.resize
  Lower:            f.lower
  Raise:            f.raise
}

menu = "EXTENDED WINDOW OPS" {
  Create Window:    .          !"xterm &"
  Iconify at New Position:  f.lowericonify
  Focus Keyboard on Window: f.focus
}
```

```
Freeze All Windows:      f.pause
Unfreeze All Windows:    f.continue
Circulate Windows Up:    f.circleup
Circulate Windows Down:  f.circledown
}
```

RESTRICTIONS

The color specifications have no effect on a monochrome system.

FILES

```
/usr/lib/X11/uwm/system.uwmrc
$HOME/.uwmrc
```

SEE ALSO

X(1), Xserver(1), xset(1), xlsfonts(1)

COPYRIGHT

COPYRIGHT 1985, 1986, 1987, 1988
DIGITAL EQUIPMENT CORPORATION
MAYNARD, MASSACHUSETTS

ALL RIGHTS RESERVED.

THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT CORPORATION. DIGITAL MAKES NO REPRESENTATIONS ABOUT THE SUITABILITY OF THIS SOFTWARE FOR ANY PURPOSE. IT IS SUPPLIED "AS IS" WITHOUT EXPRESS OR IMPLIED WARRANTY.

IF THE SOFTWARE IS MODIFIED IN A MANNER CREATING DERIVATIVE COPYRIGHT RIGHTS, APPROPRIATE LEGENDS MAY BE PLACED ON THE DERIVATIVE WORK IN ADDITION TO THAT SET FORTH ABOVE.

Permission to use, copy, modify, and distribute this software and its documentation for any purpose and without fee is hereby granted, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation, and that the name of Digital Equipment Corporation not be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission.

AUTHOR

M. Gancarz, DEC Ultrix Engineering Group, Merrimack, New Hampshire, using some algorithms originally by Bob Scheifler, MIT Laboratory for Computer Science.

NAME

val – validate SCCS file

SYNOPSIS

val –
val [**-s**] [**-rSID**] [**-mname**] [**-ytype**] *files*

DESCRIPTION

val determines if the specified *file* is an SCCS file that meets the characteristics specified by the optional argument list. Options may appear in any order, as long as they are listed before filenames. The effect of any option applies independently to each named file on the command line.

val uses the dash (–) as a special argument, causing the standard input to be read until an end-of-file condition is detected. Each line read is independently processed as if it were a command line argument list.

val generates diagnostic messages on the standard output for each command line and file processed, and also returns a single 8-bit code upon exit as described below.

The 8-bit code returned by **val** is a disjunction of the possible errors, i.e., it can be interpreted as a bit string where (moving from left to right) set bits are interpreted as follows:

- bit 0 = missing file argument
- bit 1 = unknown or duplicate keyletter argument
- bit 2 = corrupted SCCS file
- bit 3 = cannot open file or file not SCCS
- bit 4 = SID is invalid or ambiguous
- bit 5 = SID does not exist
- bit 6 = %Y%, –y mismatch
- bit 7 = %M%, –m mismatch

Note that **val** can process two or more files on a given command line and in turn can process multiple command lines (when reading the standard input). In these cases, an aggregate code is returned: a logical OR of the codes generated for each command line and file processed.

OPTIONS

- s** Silence the diagnostic message normally generated on the standard output for any error detected while processing each named file on a given command line.
- rSID** Check to see whether the value of *SID* (an SCCS delta number) is ambiguous or invalid. For example, **r1** is ambiguous because it physically does not exist but implies 1.1, 1.2, etc., which may exist. Furthermore, **r1.0** or **r1.1.0** are invalid because neither case can exist as a valid delta number. If the *SID* is valid and not ambiguous, this option also checks to see if it actually exists.
- mname** Compare the value of *name* with the SCCS %M% keyword in *file*.

-ytype Compare the value of *type* with the SCCS %Y% keyword in *file*.

BUGS

val can process up to 50 files on a single command line. Any number above 50 produces a core dump.

DIAGNOSTICS

Use **help(1)** for explanations.

SEE ALSO

admin (1), **delta (1)**, **get (1)**, **help (1)**, **prs (1)**

NAME

vc – version control

SYNOPSIS

vc [**-a**] [**-t**] [**-cchar**] [**-s**] [*keyword=value ... keyword=value*]

DESCRIPTION

vc copies lines from the standard input to the standard output under control of its *arguments* and *control statements* encountered in the standard input. In the process of performing the copy operation, user declared *keywords* may be replaced by their string *value* when they appear in plain text and/or control statements.

The copying of lines from the standard input to the standard output is conditional, based on tests (in control statements) of keyword values specified in control statements or as **vc** command arguments.

A control statement is a single line beginning with a control character, except as modified by the **-t** keyletter (see below). The default control character is colon (:), except as modified by the **-c** keyletter (see below). Input lines beginning with a backslash (\) followed by a control character are not control lines and are copied to the standard output with the backslash removed. Lines beginning with a backslash followed by a non-control character are copied in their entirety.

A keyword is composed of 9 or less alphanumeric; the first must be alphabetic. A value is any ASCII string that can be created with **ed**(1); a numeric value is an unsigned string of digits. Keyword values may not contain blanks or tabs.

Replacement of keywords by values is done whenever a keyword surrounded by control characters is encountered on a version control statement. The **-a** keyletter (see below) forces replacement of keywords in *all* lines of text. An uninterpreted control character may be included in a value by preceding it with \. If a literal \ is desired, then it too must be preceded by \.

Keyletter Arguments

- a** Forces replacement of keywords surrounded by control characters with their assigned value in *all* text lines and not just in **vc** statements.
- t** All characters from the beginning of a line up to and including the first *tab* character are ignored for the purpose of detecting a control statement. If one is found, all characters up to and including the *tab* are discarded.
- cchar** Specifies a control character to be used in place of :.
- s** Silences warning messages (not error) that are normally printed on the diagnostic output.

Version Control Statements

:dcl *keyword* [, ..., *keyword*]

Used to declare keywords. All keywords must be declared.

:asg *keyword*=*value*

Used to assign values to keywords. An **asg** statement overrides the assignment for the corresponding keyword on the **vc** command line and all previous **asg**'s for that keyword. Keywords declared, but not assigned values have null values.

:if *condition*

⋮

:end

Used to skip lines of the standard input. If the condition is true all lines between the **if** statement and the matching **end** statement are copied to the standard output. If the condition is false, all intervening lines are discarded, including control statements. Note that intervening **if** statements and matching **end** statements are recognized solely for the purpose of maintaining the proper **if-end** matching.

The syntax of a condition is:

<cond>	::= ["not"] <or>
<or>	::= <and> <and> "!" <or>
<and>	::= <exp> <exp> "&" <and>
<exp>	::= "(" <or> ")" <value> <op> <value>
<op>	::= "=" "!=" "<" ">"
<value>	::= <arbitrary ASCII string> <numeric string>

The available operators and their meanings are:

=	equal
!=	not equal
&	and
	or
>	greater than
<	less than
()	used for logical groupings
not	may only occur immediately after the <i>if</i> , and when present, inverts the value of the entire condition

The `>` and `<` operate only on unsigned integer values (e.g., `: 012 > 12` is false). All other operators take strings as arguments (e.g., `: 012 != 12` is true). The precedence of the operators (from highest to lowest) is:

```

    = != > <    all of equal precedence
    &
    |

```

Parentheses may be used to alter the order of precedence. Values must be separated from operators or parentheses by at least one blank or tab.

::text

Used for keyword replacement on lines that are copied to the standard output. The two leading control characters are removed, and keywords surrounded by control characters in text are replaced by their value before the line is copied to the output file. This action is independent of the `-a` keyletter.

:on

:off

Turn on or off keyword replacement on all lines.

:ctl char

Change the control character to char.

:msg message

Prints the given message on the diagnostic output.

:err message

Prints the given message followed by:

ERROR: *err statement on line ...* (915)

on the diagnostic output. **vc** halts execution, and returns an exit code of 1.

EXIT CODES

0 – normal

1 – any error

NAME

ver – change the version of shell commands

SYNOPSIS

ver
ver *systype*
ver *systype command*

DESCRIPTION

ver changes, temporarily or permanently, the UNIX version of commands that are executed by the shell. The command also displays the version in use.

The first form of **ver** (without arguments), displays the current value of the **SYS-
TYPE** variable, which specifies the UNIX version of commands executed by the
shell. The second form allows you to set the UNIX version to the specified *systype*.
Valid *systype* values are **sys5.3** and **bsd4.3**.

The third form of **ver** causes the UNIX version of *command* specified by *systype* to
be executed, with no permanent change to **SYSTYPE**'s value.

SEE ALSO

systype(1)

NAME

vgrind – grind nice listings of programs

SYNOPSIS

```
vgrind [ -f ] [ - ] [ -t ] [ -n ] [ -x ] [ -W ] [ -sn ] [ -h header ] [ -d file ]
[ -llanguage ] name ...
```

DESCRIPTION

vgrind formats the program sources (specified as arguments) using **troff**(1). Comments are placed in *italics*, keywords in **bold face**, and the name of the current function is listed down the margin of each page as it is encountered.

vgrind runs in two basic modes, filter mode or regular mode. In filter mode **vgrind** acts as a filter in a manner similar to **tbl**(1). The standard input is passed directly to the standard output except for lines bracketed by the following **troff**-like macros:

```
.vS      Starts processing
.vE      Ends processing
```

These lines are formatted as described above. The output from this filter can be passed to **troff** for output. There need be no particular ordering with **eqn**(1) or **tbl**(1).

In regular mode **vgrind** accepts input files, processes them, and passes them to **troff**(1) for output.

In both modes **vgrind** passes any lines beginning with a decimal point without conversion.

OPTIONS

```
-f      Forces filter mode
-       Forces input to be taken from standard input (default if -f is specified )
-t     Causes formatted text to go to the standard output (similar to the same
option in troff)
-n     Forces no keyword bolding
-x     Outputs the index file in a "pretty" format. The index file itself is produced
whenever vgrind is run with a file called index in the current directory. The
index of function definitions can then be run off by giving vgrind the -x
option and the file index as argument.
-W     Forces output to the (wide) Versatec printer rather than the (narrow) Varian
(BSD troff doesn't support the Versatec).
-s     Specifies a point size to use on output (exactly the same as the argument of a
.ps)
-h     Specifies a particular header to put on every output page (the default is the
filename)
-d     Specifies an alternate language definitions file (the default is
/usr/lib/vgrindefs)
```

- l Specifies the language to use. Currently known are PASCAL (-lp), MODEL (-lm), C (-lc or the default), CSH (-lcsh), SHELL (-lsh), RATTOR (-lr), MODULA2 (-lmod2), YACC (-lyacc), ISP (-lisp), and ICON (-li).

FILES

index	file where source for index is created
/usr/lib/tmac/tmac.vgrind	macro package
/usr/lib/vfontedpr	preprocessor
/usr/lib/vgrindefs	language descriptions

BUGS

vfontedpr assumes that a certain programming style is followed:

For **C** – function names can be preceded on a line only by spaces, tabs, or an asterisk. The parenthesized arguments must also be on the same line.

For **PASCAL** – function names need to appear on the same line as the keywords **function** or **procedure**.

For **MODEL** – function names need to appear on the same line as the keywords **is** or **beginproc**.

If these conventions are not followed, the indexing and marginal function name comment mechanisms will fail.

More generally, arbitrary formatting styles for programs mostly look bad. The use of spaces to align source code fails miserably; if you plan to **vgrind** your program you should use tabs. This is somewhat inevitable since the font used by **vgrind** is variable width.

The mechanism of **ctags** in recognizing functions should be used here.

The filter mode does not work in documents using the **-me** or **-ms** macros.

SEE ALSO

vgrindefs(5)

NAME

vi – screen oriented (visual) display editor based on **ex**

SYNOPSIS

vi [**-t** *tag*] [**-r** *file*] [**+command**] [**-l**] [**-wn**] *name* ...

DESCRIPTION

vi is a display-oriented text editor based on **ex**(1). Since **vi** runs the same code as **ex**(1), you can get to the command mode of **ex**(1) from within **vi** and vice versa.

When using **vi**, changes you make to the file are reflected in what appears on the terminal screen. The position of the cursor on the screen indicates the position within the file. The *UNIX Text Processing* manual provides full details on the use of **vi**.

OPTIONS

-t <i>tag</i>	Edit the file containing the <i>tag</i> and position the editor at its definition.
-r <i>file</i>	Recover <i>file</i> after an editor or system crash. If <i>file</i> is not specified, a list of all saved files is printed.
-l	Indent appropriately for LISP code, and modify (, {), [, and] commands to have meaning for LISP.
-wn	Set the default window size to <i>n</i> .
+command	Force the editor to begin by executing the specified <i>command</i> . If you omit <i>command</i> , vi initially positions the cursor at the last line of the first file.

NOTES

In the Domain/OS BSD implementation of **vi**, the **w** (word) and **^]** (tag) commands both recognize a **\$** as part of a word. The **w** command also recognizes a **-** as part of a word, if LISP mode is on.

BUGS

Software tabs using CTRL/T work only immediately after the *autoindent*.

Left and right shifts on intelligent terminals don't make use of insert and delete character operations in the terminal.

The *wrapmargin* option can be fooled since it looks at output columns when blanks are typed. If a long word passes through the margin and onto the next line without a break, then the line won't be broken.

Wrapmargins should be from left, not right, margin.

Insert/delete within a line can be slow if tabs are present on intelligent terminals, since the terminals need help in doing this correctly.

Saving text on deletes in the named buffers is somewhat inefficient.

The *source* command does not work when executed as **:source**; there is no way to use the **:append**, **:change**, and **:insert** commands, since it is not possible to give more than one line of input to a colon (:) escape. To use these on a **:global** you must **Q** to **ex** command mode, execute them, and then reenter the screen editor with **vi** or **open**.

FILES

<code>/usr/lib/ex?.?strings</code>	Error messages
<code>/usr/lib/ex?.?recover</code>	Recover command
<code>/usr/lib/ex?.?preserve</code>	Preserve command
<code>/etc/termcap</code>	Terminal capability descriptions
<code>~/exrc</code>	Editor start-up file
<code>/tmp/Exnnnnnn</code>	Editor temporary
<code>/tmp/Rxnnnnnn</code>	Named buffer temporary
<code>/usr/preserve</code>	Preservation directory

SEE ALSO

`ex(1)`, `edit(1)`, `termcap(5)`, `environ(7)`,
UNIX Text Processing.

NAME

vsize – set/display VT100 window settings

SYNOPSIS

vsize [*options*]

DESCRIPTION

The **vsiz**e command allows you to set the dimensions of the VT100 emulator window pane. This command is valid only from within the VT100 emulator (which is invoked with the **VT100** command); attempting to use it directly from the shell causes an error.

OPTIONS

If no options are specified, **vsiz**e displays the current window pane settings.

- l n** Specifies the height of the window pane in lines. If this option is omitted, the height remains unchanged.
- c n** Specifies the width of the window in columns. If this option is omitted, the width remains unchanged.
- std** Sets the height of the window to 24 lines and the width to 80 columns. This is the same as saying **-l 24 -c 80**.

EXAMPLES

Invoke VT100 emulator and Display current settings.

```
$ vt100
$ vsiz
Screen size is 18 lines by 70 columns.
```

Change the width. Exit the emulator and return to the shell.

```
$ vsiz -c 60
Old screen size is 18 lines by 70 columns.
New screen size is 18 lines by 60 columns.
$ *** EOF ***
$
```

NAME

vt100 – VT100 terminal emulator

SYNOPSIS

vt100 [*options*] [*pathname* [*arg1* *arg2* ...]]

DESCRIPTION

The **vt100** command creates a window running the VT100 terminal emulator and starts up a shell within the window.

The VT100 terminal emulation package is intended for use with two types of programs. When used in conjunction with remote communications packages such as Domain TCP/IP or X.25, the VT100 terminal emulator allows you to interact with the remote system as if you were logged into a VT100 connected to that system. Using the VT100 terminal emulator with programs that take advantage of VT100 special features allows you to run these programs on a Domain node without having to tailor them to the Domain environment.

pathname [*arg1* *arg2* ...] (optional)

Specify the name of a command or program for the shell in the VT100 window to invoke. You must give the full pathname; for example, */com/ld*. *arg1*, *arg2*, ... are valid arguments to the selected command (or program): for example, */com/ld //my_node/my_home_dir*. The default is to invoke the value of the variable *\$\$SHELL*, or if *\$\$SHELL* is not set, invoke */com/sh*.

OPTIONS

If any options are specified, they must precede the argument(s). Once **vt100** is running, you may change the window size with the **vsiz** command.

- std** Set up a VT100 window that is 24 lines by 80 columns (the standard size of a VT100 screen).
- lines *n*** Set up a VT100 window with the number of lines specified by *n*. The number of lines cannot exceed the number of lines in the DM window running the VT100 emulator.
- columns *n*** Set up a VT100 window with the number of columns specified by *n*. The number of columns cannot exceed the number of columns of the DM window running the VT100 emulator.

The VT100 terminal emulation package consists of the following:

- The terminal emulation software, which performs the functions of a VT100 terminal, such as handling VT100-type escape sequences. The terminal emulator redirects the handling of keyboard input and screen output to stream manager operations. The terminal emulator is invoked within a DM window by the **vt100** shell command.
- The terminal emulator driver, which performs keyboard input functions such as erasing or echoing characters.

EXAMPLES

1. Create a window running the VT100 emulator and start a shell running within the window.

\$ vt100

2. Open a connection to the remote system specified by *hostname* and create a window running the VT100 emulator.

\$ vt100 login hostname

KEYBOARD LAYOUT

The table below shows how the keys on a Domain low-profile keyboard map to the keys of a VT100. This assumes that you are running the VT100 Keyboard Emulation package on your node. Note that the VT100 definitions for the F2, F3, and F7 keys supersede the usual **emt** definitions for these keys.

Domain key	Vt100 keypad
<INS MODE>	<ESC>
<CHAR DEL>	<RUBOUT>
<F2>	<PF1>
<F3>	<PF2>
<F4>	<PF3>
<F5>	<PF4>
SHIFT/<F2>	<7>
SHIFT/<F3>	<8>
SHIFT/<F4>	<9>
SHIFT/<F5>	<->
CTRL/<F2>	<4>
CTRL/<F3>	<5>
CTRL/<F4>	<6>
CTRL/<F5>	<,>
<F6>	<1>
<F7>	<2>
SHIFT/<F6>	<3>
SHIFT/<F7>	<ENTER>
CTRL/<F6>	<0>
CTRL/<F7>	<->

NAME

wait – await completion of process

SYNOPSIS

wait

DESCRIPTION

wait pauses until all processes started with an ampersand (&) have completed execution. It then reports on abnormal terminations.

Because the **wait(2)** system call must be executed in the parent process, the shell itself executes **wait** without creating a new process.

BUGS

Not all the processes of a three-stage (or longer) pipeline are children of the shell **sh(1)**. Thus, you may not be able to **wait** for all processes. This problem does not apply to **csh(1)**.

SEE ALSO

sh(1), **ksh(1)**

NAME

wall – write to all users

SYNOPSIS

wall

DESCRIPTION

wall reads its standard input until an end-of-file. It then sends this message, preceded by "Broadcast Message ...", to users logged-in on the node from which you are writing. The message appears in a pad on the transmitting node, as well as on the screen or output device of anyone logged onto the node via ttys or SIO lines.

The sender should be super-user to override any protections the users may have invoked.

DIAGNOSTICS

"Cannot send to ..." when the open on a user's tty or sio file fails.

FILES

/dev/tty?
/etc/utmp
/dev/pty?
/dev/sio?

SEE ALSO

mesg(1), write(1)

NAME

wbak – create a magnetic media backup file

SYNOPSIS

```
wbak -f fileno [-dev | m[unit] | f | ct]
      [-full|-incr|-af dtm|-bef dtm]
      [-fid id] [-own id] [-vid vol_id]
      [-sla|-nsla] [-wla|-nwla] [-nhi] [-pdtu]
      [-reo] [-reten|-nreten] [-no_eot]
      [-sysboot] [-l|-ld|-lf|-ll] [-to filename]
      [-type uasc|unstruct|hdru]
      [-r] [-stdout] [-presr10] pathname...
```

DESCRIPTION

wbak writes one or more objects to either a removable media, disk file or standard output. These objects may be directory trees, files, or links. For each object, the information saved includes the name, object data, and attributes associated with the object, such as the access control list. This lets you reconstruct files, the directory tree, or any portion of the tree using the **rbak** command.

The **wbak** and **rbak** commands are intended both for disk backup and for interchanging information between separate Apollo installations. Use the **rwmt** command to read and write magnetic media that are used for interchanging information with non-Apollo installations.

pathname (required)

Specify the name of the object to be written to backup media. This may be a directory, file, or link. If it is a file, then the file is written as specified. If it is a link, then the link is resolved and the resolution object is written to backup media. If it is a directory, all subordinate files and subdirectories in the tree are written. Note that the *pathname* specified reflects the way the directory is stored on the backup media, and that the same name must be used when reading files using *pathnames* in **rbak**. Multiple *pathnames* and wildcarding are permitted. If you omit this argument, **wbak** will prompt you for it. You may specify a hyphen (-) as an argument to direct **wbak** to standard input for further arguments and options.

OPTIONS

The **-f** option is required, as it specifies where on the backup media the new file is to be written. If you omit it, **wbak** will prompt you for it.

Tape File Identifiers

-fid *file_id*

Specify a 1-17 character file ID to be written in the file header label for use when writing a file to a labeled volume. If this option is omitted, the file is not named and can only be restored by the file number.

-f [*position*]

Specify the file position for the write operation. Valid values for *position* are **cur**, **end**, or a nonzero integer. A position of **cur** specifies that the file should be written at the current position on the backup media; the media must have been previously written by **wbak** and its position must not have been disturbed.

A position of **end** specifies that the file should be written at the end of the backup media file set. This causes **wbak** to append the specified disk file (*pathname* argument) to the very end of the file set.

A position specified by a nonzero integer value causes the file to be written at that absolute position in the backup media volume. If multiple *pathname* arguments are supplied, the value of *position* is incremented by one after each file has been written.

The default value for *position* is 1.

Mode Control

The object specified by the *pathname* argument must be a directory for either **-full** or **-incr** to have meaning.

-full (default)

Specify a full backup; save all files in specified trees.

-incr

Specify an incremental backup; save files that were modified since the last backup recorded in the **backup_history** file stored in the *pathname* directory.

-af *dtm*

Save all files modified after the given date and time; *dtm* is in the form *yy/mm/dd.hh:mm*. The date defaults to today, and the time to midnight if either of those are omitted from *dtm*.

-bef *dtm*

Save all files last modified before the given date and time.

Label Control

-wla (default)

Write the backup media volume label if the backup file number is 1.

-nwla

Suppress writing of the backup media volume label.

-own *id*

Specify backup media volume owner (1–14 character name). This option is only meaningful when used with the **-wla** option.

-vid *vol_id*

Specify a 1–6 character volume ID for use when labeling a volume. This option is only meaningful when the backup file number is 1. The default volume ID is ' ' (blank).

- sla** (default) Display the label information written for this backup file on standard output.
- nsla** Suppress output of label information.

Listing Control

You may include the **-l** option, or any combination of **-ld**, **-lf**, and **-ll**.

- l** Write the names of all files, directories, and links saved to standard output.
- lf** Write the names of all files saved to standard output.
- ld** Write the names of all directories saved to standard output.
- ll** Write the names of all links saved to standard output.

Backup Device Control

- dev *d[unit]*** Specify device type and unit number. *d* must be either **m** (for reel-to-reel magnetic tape), **ct** (for cartridge tape), or **f** (for floppy), depending on which drive is being used. *unit* is an integer (0-3). Both are required for reel-to-reel tapes (that is, **-dev m2**). A unit number is not required for floppy disks and cartridge tapes (that is, **-dev f**). If this option is omitted, **rbak** assumes device **m0**.

CAUTION: Floppy disk support for this command is limited. In particular, error detection during reads and writes is poor. do not use this command with floppy disks when the data being placed on the floppy disks are critical and unrecoverable.

- to *filename*** Backup output is written to the specified streams object rather than removable media. This can then be restored using the **-from** option in **rbak**. If the file already exists, use the **-r** option to replace it. If **-type** option is not specified the file will be assigned the default type. You cannot use the **-file *n*** option with streams.
- type [*uasc* | *unstruct* | *hdru*]** Specify the type of the object *filename*. It can be one of ASCII (**uasc**), Unstructured (**unstruct**) or Streams header-undefined (**hdru**) type.
- r** If the object specified with the **-to** option already exists, this option allows it to be replaced. The type of *filename* is however left unchanged.

- stdout** The backup output is written to standard output.
- reo** Force previous backup media volume to be reopened, and suppress reading of backup media volume label. Use only when backup media has not been repositioned since last **wbak** or **rbak**.
- unload** Unload the tape at the end of the backup session. This option is valid for reel-to-reel magnetic tapes only.
- rewind** Rewind the tape after reading or indexing. If this option is omitted, the tape is left positioned to the next tape file.

Special Cartridge Tape Control Options

- reten** Retension the cartridge tape (unwind to the end, then rewind). This can be helpful if you have encountered cartridge tape reading errors. Retensioning requires about 1.5 minutes to complete.
- nreten** (default) Do not retension the cartridge tape.
- no_eot** Suppress the writing of two tape marks at the end of the tape file, which are the standard signal for end of tape. The cartridge can't position between the two tapemarks to be ready for a successive call to **wbak** (as it does on magtape), without rewinding the tape and searching forward, so this option speeds up multiple invocations of **wbak**. It should not be used on the last invocation of **wbak**. Also, **-f cur** should be used on all **wbak** invocations in a series except the first one.
- sysboot** Permit use of a bootable tape that has a special boot program at the beginning. This option causes **wbak** to skip over the first file on the tape. This option is only necessary when the first file on the tape is being written (**-f 1**).

Miscellaneous Control Options

- nhl** Suppress update of the backup history file.
- (hyphen) Read standard input for further arguments or options; input is accepted until **wbak** receives an EOF.
- pdtu** Preserves the last date/time-used information on objects. After each object is backed up on tape, the date/time-used information is reset to the value it had before the backup.
- presr10** Allows you to make a tape with pre-SR10 format from an SR10 node. This tape will have no ACLs by default. You can restore it to a pre-SR10 volume by means of the pre-SR10 **rbak**. If you make a tape

without this option it will not be readable on a pre-SR10 system.

EXAMPLES

```
$ wbak //mask/wby -f 1 -af 87/11/19.12.00 -fid wby -L
```

This command writes the directory `//mask/wby` to tape. The directory is written out to tape file one, and the file ID `wby` is written to the file's label. Disk files from directory `wby` are written to the tape only if they have been modified since noon on November 19, 1987. The label and the names of the files written to tape are printed to standard output.

When this command is executed, `wbak` writes the following information to standard output:

```
Label:
  File number:    1
  File section:   1
  File id:        wby
  Date written:   1987/11/20 10:47:58 EST
```

Starting write:

```
{file} "//mask/wby/among" written
{file} "//mask/wby/school" written
{file} "//mask/wby/children" written
{file} "//mask/wby/backup_history" written
{dir}  "//mask/wby/" written.
```

Write complete.

This command backs up the entire contents of the node whose entry directory name is `goeey`. Note that the file ID is specified as `"node 27 backup"` to make it easy to identify when you want to reload it, and that the command assigns volume and owner IDs.

```
$ wbak -f 1 -own "john doe" -vid "volbk2" -fid "node 27 backup" //goeey
```

When this command is executed, `wbak` writes the following information to standard output:

```
Label:
  Volume id:      VOLBK2
  Owner id:       john doe
  File number:    1
  File section:   1
```

```
File id:      n 27 backup
File written: 1987/02/17 18:00:39 EST
```

Starting write:

Write complete.

This command uses wildcards to match only those files in the **ug** subdirectory of the current working directory whose names begin with the letters **a** through **f** and end with **_example**.

```
$ wbak -f 1 -own "john doe" -vid "volbk1" ug/[a-f*]_example -l
```

When this command is executed, **wbak** writes the following information to standard output:

```
Label:
Volume id:      VOLBK1
Owner id:       john doe
File number:    1
File section:   1
File id:        (no id specified)
File written:   1988/02/17 17:58:52 EST
```

Starting write:

```
(file) "ug/cmf_example" written.
(file) "ug/cmt_example" written.
(file) "ug/cpboot_example" written.
(file) "ug/cpf_example" written.
(file) "ug/cpt_example" written.
(file) "ug/fpat_example" written.
(file) "ug/fppmask_example" written.
(file) "ug/fst_example" written.
```

Write complete.

```
$ wbak src -to /backup/bck_out.file
```

This command writes the backup output for the directory **src** to the file **/fred/bck_out.file**. The directory can be restored in either of the following two ways :

```
rbak src -from /backup/bck_out.file
```

or

```
cat /fred/bck_out.file | rbak src -stdin
```

Using streams as a backup output media, it is possible to stage the backup output to intermediate disks and then use **rwmt** to write the intermediate file to the magnetic tape. The sequence to use is as follows

```
$ wbak //otter -to //backup/ot wbak //otter -to //backup/tmp1
```

This writes the backup output to an intermediate file `//backup/tmp1` followed by

```
rwmt -f 2 -w //backup/tmp1 -raw -rl 8192 -nobs -ansi
```

When the magtape unit is available at a later time the intermediate file is written to the magtape. Note that it is **ESSENTIAL** to use the **-raw**, **-rl 8192** and the **-nobs** options of **rwmt**, for **rbak** to be able to read the backup from tape. All tapes used for this must have the ANSI specified volume label. You can only use this sequence for magnetic tapes. **rbak** will not be able to restore data written using the above sequence for cartridge tapes instead of magnetic tapes. This sequence has exactly the same effect as using

```
wbak //otter -dev mt -f 2
```

You can then use **rbak** as follows to retrieve the data

```
rbak //otter -f 2 -dev mt
```

SEE ALSO

rbak(1), **rwmt(1)**

NAME

wc – word count

SYNOPSIS

wc [**-lwc**] [*name* ...]

DESCRIPTION

wc counts lines, words and characters in the named files, or in the standard input if no filename appears. A word is defined as a maximal string of characters delimited by spaces, tabs or newlines.

If you provide more than one filename as an argument, **wc** performs an individual count on each file, as well as a total count of lines, words and characters in all the files. In any case, **wc** lists the specified *names* along with the count.

OPTIONS

The options can be combined to select which counts will be reported.

- l** Report a line count.
- w** Report a word count.
- c** Report a character count.

EXAMPLES

```
% wc namelist
61      127      853 namelist
```

In the previous example, the first column indicates the number of lines, followed by the number of words, the number of characters, and finally, the filename.

The following example shows the individual line and word count for the files **namelist** and **namelist.update**, and the total number of lines and words in the two files.

```
% wc -lw namelist namelist.update
61      127 namelist
270     447 namelist.update
331     574 total
```

NAME

what – show what versions of object modules were used to construct a file

SYNOPSIS

what *file* ...

DESCRIPTION

what reads each file and searches for sequences of the form "@(#)" as inserted by the source code control system. It then prints the remainder of the string after this marker, up to a null character, newline, double quote, or ">" character.

SEE ALSO

sccs(1)

NAME

whatis – describe what a command is

SYNOPSIS

whatis *command* ...

DESCRIPTION

whatis looks up a given command, system call, library function, or special file name, and gives the header line from the manual section. This header line should contain the section number and command name, along with a very brief one-line description. You can then run the **man**(1) command to get more information about the particular command. If the line begins with *name(section)* ... you can use **man** *section name* to get the documentation for it.

whatis is actually just the **-f** option to the **man**(1) command.

This command is only useful if you know the name of what you're looking for. If you're not sure, **apropos**(1) is more thorough about searching the header lines, which sometimes aren't complete descriptions of the command's purpose.

EXAMPLE

```
% whatis ed
ed(1) - text editor
```

You can type the following line to get the manual entry for **ed**.

```
man 1 ed
```

FILES

/usr/man/whatis Data base

SEE ALSO

apropos(1), **man**(1), **catman**(8)

NAME

whereis - locate source, binary, and/or manual for program

SYNOPSIS

whereis [**-sbm**] [**-u**] [**-SBM dir ... -f**] *name* ...

DESCRIPTION

whereis locates source/binary and manuals sections for specified files. The supplied names are first stripped of leading pathname components and any single extension consisting of a period followed by one or more characters, e.g., **.c**. **whereis** also ignores prefixes of "s." resulting from use of source code control (SCCS). **whereis** attempts to locate the desired program in a list of standard places.

OPTIONS

The **-s**, **-b**, and **-m** options can be combined.

- s** Search for sources.
- b** Search for binaries.
- m** Search for manual sections.
- u** Search for unusual entries. A file is said to be unusual if it does not have one entry of each requested type. Thus, **whereis -m -u *** asks for those files in the current directory which have no documentation.
- B dir ...** Change or otherwise limit search for binaries to the specified directories.
- M dir ...** Change or otherwise limit search for manual sections to the specified directories.
- S dir ...** Change or otherwise limit search for sources to the specified directories.
- f** Terminate the last directory list and signal the start of file names.

EXAMPLE

The following finds all the files in **/usr/ucb** which are not documented in **/usr/man/man1** with source in **/usr/src/cmd**:

```
cd /usr/ucb
whereis -u -M /usr/man/man1 -S /usr/src/cmd
```

FILES

```
/usr/src/*
/usr/{doc,man}/*
/lib, /etc, /usr/{lib,bin,ucb,old,new,local}
```

BUGS

Since the program uses **chdir(2)** to run faster, pathnames given with the options **-M**, **-S**, and **-B** must be full, i.e., they must begin with a slash.

SEE ALSO

which(1)

NAME

which – locate a program file, including aliases and paths (**csh** only)

SYNOPSIS

which [*names*]

DESCRIPTION

which takes a list of *names* and looks for the files that would be executed if these names were given as commands. It expands arguments that are aliased, and searches for them along your path.

DIAGNOSTICS

which provides a diagnostic for names aliased to more than a single word, or for executable files whose argument names are not found in the path.

BUGS

A C shell must execute **which**, since other shells do not know about aliases. Refer to **csh(1)** for more information.

SEE ALSO

csh(1)

NAME

who – who is on the system

SYNOPSIS

```
who [ -w ] [ -a | -d | -f nodefile | -n arg ] [ who-file ]
who [ am i ]
```

DESCRIPTION

who, without any options or arguments, lists the log-in name, terminal name, and log-in time for each user currently logged in.

Without an argument, **who** examines the */etc/utmp* file to obtain its information. If you specify a file, that file is used instead of */etc/utmp*. Typically the specified file is */usr/adm/wtmp*, which contains a record of all the logins since it was created. Then **who** lists logins, logouts, and crashes since the creation of the *wtmp* file. Each login is listed with username, terminal name (with */dev/* suppressed), and date and time. When an argument is given, logouts produce a similar line without a username. Reboots produce a line with *x* in place of the device name, and a fossil time indicative of when the system went down.

who am i (or **who** with two arguments) tells who you are logged in as.

Domain/OS BSD OPTIONS

- | | |
|--------------------------|--|
| -w | Produces wide output. Normally, lines are printed with the name and node fields 20 characters wide. The -w option prints the maximum name and node field widths. |
| -a | Processes <i>/etc/utmp</i> or the named <i>file</i> for all nodes in the ring. |
| -d | Displays information about all nodes booted off the current boot volume. This can be useful for listing users who should be notified of an impending system shutdown. |
| -f nodefile | Specifies nodes for which <i>/etc/utmp</i> will be processed. <i>nodefile</i> should contain lines in the form: <i>//nodename</i> or <i>[net.]nodeid</i> , one per line. A pound sign (#) in the first column causes that line to be treated as a comment. |
| -n arg1[,arg2...] | Specifies nodes for which <i>/etc/utmp</i> will be processed. The <i>arg</i> lists the nodes to be processed, and should be in the form: <i>//nodename</i> or <i>[net.]nodeid</i> . If more than one node is specified, they should be either separated by commas or separated by whitespace, and the entire argument should be in quotes. |

You can use only one of **-a**, **-d**, **-f**, or **-n**.

FILES

/etc/utmp

SEE ALSO

last(1), *whoami(1)*, *getuid(2)*, *utmp(5)*

NAME

whoami – print effective current user id

SYNOPSIS

whoami

DESCRIPTION

whoami prints who you are. It works even if you are su'd, while **who am i** does not since it uses */etc/utmp*.

FILES

/etc/passwd Name database

SEE ALSO

who(1)

NAME

whois – DARPA Internet username directory service

SYNOPSIS

whois *name*

DESCRIPTION

Entering the command **whois help** produces a helpful message similar to the following:

Please enter a name or a handle ("ident") such as "Smith" or "SRI-NIC". Starting with a period forces a name-only search; starting with an exclamation point forces handle-only. Examples:

Smith	[looks for name or handle SMITH]
!SRI-NIC	[looks for handle SRI-NIC only]
.Smith, John	[looks for name JOHN SMITH only]

Adding "..." to the argument matches anything from that point, e.g. "ZU..." matches ZUL, ZUM, etc.

To have the entire membership list of a group or organization, shown with the record, use an asterisk (*) directly preceding the given argument. You can, of course, use an exclamation point and asterisk, or a period and asterisk together.

NAME

write – write to another user

SYNOPSIS

write *user* [*ttyname*]

DESCRIPTION

write copies lines from your terminal to that of another user. When first called, it sends the message

Message from yourname@yoursystem on yourttyname at time...

The recipient of the message should write back at this point. Communication continues until an end of file is read from the terminal or an interrupt is sent. At that point **write** writes 'EOT' on the other terminal and exits.

If you want to write to a user who is logged in more than once, the *ttyname* argument may be used to indicate the appropriate terminal name.

Permission to write may be denied or granted by use of the **mesg** command. At the outset writing is allowed. Certain commands, in particular **nroff** and **pr(1)**, disallow messages in order to prevent messy output.

If the character '!' is found at the beginning of a line, **write** calls the shell to execute the rest of the line as a command.

The following protocol is suggested for using **write**: when you first write to another user, wait for him to write back before starting to send. Each party should end each message with a distinctive signal—(o) for 'over' is conventional—that the other may reply. The signal (oo) for 'over and out' is suggested when conversation is about to be terminated.

FILES

/etc/utmp	to find user
/bin/sh	to execute '!'

SEE ALSO

mesg(1), **who(1)**, **mail(1)**

NAME

X - a portable, network transparent window system

SYNOPSIS

X is a network transparent window system developed at MIT which runs on a wide range of computing and graphics machines. The core distribution from MIT has both client and server support for the following operating systems:

4.3+tahoe
 Ultrix 3.0 FT2 (also compiles under Ultrix 2.0)
 SunOS 3.4
 HP-UX 6.01
 Apollo Domain/IX 9.7
 Apollo Domain/OS 10.2 (and 10.1)
 IBM AOS 4.3 (according to its developers)
 A/UX 1.0

Commercial implementations are also available for a much wider range of platforms.

The X Consortium requests that the following names be used when referring to this software:

X
 X Window System
 X Version 11
 X Window System, Version 11
 X11

X Window System is a trademark of the Massachusetts Institute of Technology.

DESCRIPTION

X window system servers run on computers with bitmap displays. The server distributes user input to and accepts output requests from various client programs through a variety of different interprocess communication channels. Although the most common case is for the client programs to be running on the same machine as the server, clients can be run transparently from other machines (including machines with different architectures and operating systems) as well.

X supports overlapping hierarchical subwindows and text and graphics operations, on both monochrome and color displays. For a full explanation of the functions that are available, see the *Xlib - C Language X Interface* manual, the *X Window System Protocol* specification, and various toolkit documents.

The number of programs that use X is growing rapidly. Of particular interest are: a terminal emulator (*xterm*), a window manager (*uwm*), a display manager (*xm*), mail managing utilities (*xmh* and *xbiff*), a manual page browser (*xman*), a bitmap editor (*bitmap*), an access control program (*xhost*), user preference setting programs (*xrdb*, *xset*, *xsetroot*, and *xmodmap*), a load monitor (*xload*), clock (*xclock*), a font display (*xfd*), utilities for listing information about fonts, windows, and displays (*xlsfonts*,

xlswins, *xwininfo*, *xdpyinfo*, and *xprop*), a protocol translator for running X10 programs (*x10tox11*), a diagnostic for seeing what events are generated and when (*xev*), screen image manipulation utilities (*xwd*, *xwud*, *xpr*, and *xmag*), and various demos (*xeyes*, *ico*, *muncher*, *puzzle*, etc.).

Many other utilities, window managers, games, toolkits, etc. are available from the user-contributed distribution. See your site administrator for details.

STARTING UP

There are currently 3 ways of getting the X server and an initial set of client applications started. The particular method used depends on what operating system you are running and on whether or not you use other window systems in addition to X.

xdm (the X Display Manager)

Applicable for Xapollo when running in borrow mode.

If you want to always have X running on your display, your site administrator can set your machine up to use the X Display Manager *xdm*. This program is typically started by the system at boot time and takes care of keeping the server running and getting users logged in. If you are running *xdm*, you will see a window on the screen welcoming you to the system and asking for your username and password. Simply type them in as you would at a normal terminal, pressing the Return key after each. If you make a mistake, *xdm* will display an error message and ask you to try again. After you have successfully logged in, *xdm* will start up your X environment. By default, if you have an executable file named *xsession* in your home directory, *xdm* will treat it as a program (or shell script) to run to start up your initial clients (such as terminal emulators, clocks, a window manager, user settings for things like the background, the speed of the pointer, etc.). Your site administrator can provide details.

xterm -L (started from */etc/init*)

Some versions of UNIX that are derived from 4.3bsd support starting the window system and an initial *xterm* window from the system terminal line configuration file */etc/tty*s. As with *xdm*, there will be a window requesting your username and password. However, this window will become your primary window and is not configurable on a per-user basis. Sites using this method should switch to *xdm* as *xterm -L* may not be supported in future releases.

xinit (run manually from the shell)

Sites that support more than one window system might choose to use the *xinit* program for starting X manually. If this is true for your machine, your site administrator will probably have provided a program named "x11", "startx", or "xstart" that will do site-specific initialization (such as loading convenient default resources, running a window manager, displaying a clock, and starting several terminal emulators) in a nice way. If not, you can build such a script using the *xinit* program. This utility simply runs one user-specified program to start the server, runs another to start up any desired clients, and then waits for either to finish. Since either or both of

the user-specified programs may be a shell script, this gives substantial flexibility at the expense of a nice interface. For this reason, *xinit* is not intended for end users.

DISPLAY NAMES

From the user's prospective, every X server has a *display name* of the form:

hostname:displaynumber.screennumber

This information is used by the application to determine how it should connect to the server and which screen it should use by default (on displays with multiple monitors):

hostname

The *hostname* specifies the name of the machine to which the display is physically connected. If the hostname is not given, the most efficient way of communicating to a server on the same machine will be used.

displaynumber

The phrase "display" is usually used to refer to collection of monitors that share a common keyboard and pointer (mouse, tablet, etc.). Most workstations tend to only have one keyboard, and therefore, only one display. Larger, multi-user systems, however, will frequently have several displays so that more than one person can be doing graphics work at once. To avoid confusion, each display on a machine is assigned a *display number* (beginning at 0) when the X server for that display is started. The display number must always be given in a display name.

screennumber

Some displays share a single keyboard and pointer among two or more monitors. Since each monitor has its own set of windows, each screen is assigned a *screen number* (beginning at 0) when the X server for that display is started. If the screen number is not given, then screen 0 will be used.

On UNIX systems, the default display name is stored in your DISPLAY environment variable. This variable is set automatically by the *xterm* terminal emulator. However, when you log into another machine on a network, you'll need to set DISPLAY by hand to point to your display. For example,

```
% setenv DISPLAY myws:0
$ DISPLAY=myws:0; export DISPLAY
```

Finally, most X programs accept a command line option of *-display displayname* to temporarily override the contents of DISPLAY. This is most commonly used to pop windows on another person's screen or as part of a "remote shell" command to start an xterm pointing back to your display. For example,

```
% xeyes -display joesws:0 -geometry 1000x1000+0+0
% rsh big xterm -display myws:0 -ls </dev/null &
```

X servers listen for connections on a variety of different communications channels (network byte streams, shared memory, etc.). Since there can be more than one way of contacting a given server, The *hostname* part of the display name is used to determine the type of channel (also called a transport layer) to be used. The sample servers from MIT support the following types of connections:

TCP/IP

The hostname part of the display name should be the server machine's IP address name. Full Internet names, abbreviated names, and IP addresses are all allowed. For example: *expo.lcs.mit.edu:0*, *expo:0*, *18.30.0.212:0*, *bigmachine:1*, and *hydra:0.1*.

UNIX domain sockets

Use only the *displaynumber.screen number* syntax, not a host named "unix," for example, *:0*, *:0.1*.

DECnet

The hostname part of the display name should be the server machine's nodename followed by two colons instead of one. For example: *myws::0*, *big::1*, and *hydra::0.1*.

GEOMETRY SPECIFICATIONS

One of the advantages of using window systems instead of hardwired terminals is that applications don't have to be restricted to a particular size or location on the screen. Although the layout of windows on a display is controlled by the window manager that the user is running (described below), most X programs accept a command line argument of the form **-geometry WIDTHxHEIGHT+XOFF+YOFF** (where *WIDTH*, *HEIGHT*, *XOFF*, and *YOFF* are numbers) for specifying a preferred size and location for this application's main window.

The *WIDTH* and *HEIGHT* parts of the geometry specification are usually measured in either pixels or characters, depending on the application. The *XOFF* and *YOFF* parts are measured in pixels and are used to specify the distance of the window from the left or right and top and bottom edges of the screen, respectively. Both types of offsets are measured from the indicated edge of the screen to the corresponding edge of the window. The X offset may be specified in the following ways:

- +XOFF** The left edge of the window is to be placed *XOFF* pixels in from the left edge of the screen (i.e. the X coordinate of the window's origin will be *XOFF*). *XOFF* may be negative, in which case the window's left edge will be off the screen.
- XOFF** The right edge of the window is to be placed *XOFF* pixels in from the right edge of the screen. *XOFF* may be negative, in which case the window's right edge will be off the screen.

The Y offset has similar meanings:

- +YOFF** The top edge of the window is to be *YOFF* pixels below the top edge of the screen (i.e. the Y coordinate of the window's origin will be *YOFF*). *YOFF* may be negative, in which case the window's top edge will be off the screen.

-YOFF The bottom edge of the window is to be *YOFF* pixels above the bottom edge of the screen. *YOFF* may be negative, in which case the window's bottom edge will be off the screen.

Offsets must be given as pairs; in other words, in order to specify either *XOFF* or *YOFF* both must be present. Windows can be placed in the four corners of the screen using the following specifications:

- +0+0** upper left hand corner.
- 0+0** upper right hand corner.
- 0-0** lower right hand corner.
- +0-0** lower left hand corner.

In the following examples, a terminal emulator will be placed in roughly the center of the screen and a load average monitor, mailbox, and clock will be placed in the upper right hand corner:

```
xterm -fn 6x10 -geometry 80x24+30+200 &
xclock -geometry 48x48-0+0 &
xload -geometry 48x48-96+0 &
xbiff -geometry 48x48-48+0 &
```

WINDOW MANAGERS

The layout of windows on the screen is controlled by special programs called *window managers*. Although many window managers will honor geometry specifications as given, others may choose to ignore them (requiring the user to explicitly draw the window's region on the screen with the pointer, for example).

Since window managers are regular (albeit complex) client programs, a variety of different user interfaces can be built. The core distribution comes with a window manager named *uwm* which supports overlapping windows, popup menus, point-and-click or click-to-type input models, and minimal use of screen real estate (i.e. it doesn't provide title bars around windows). It tries to stay out of the way as much as possible, but can be somewhat difficult to master.

Several other window managers are available in the user-contributed distribution:

- awm*** This window manager is descended from *uwm* but provides optional title bars whose layout can be tailored by the user.
- twm*** This window manager provides title bars, resize and reposition boxes, and specially-designed icons. It is known for its small size and simplicity.
- rtl*** This is a *tiling* window manager that rearranges and resizes windows on the screen to prevent them from ever overlapping.

People who find that none of these window managers are acceptable are encouraged to write their own.

FONT NAMES

Collections of characters for displaying text and symbols in X are known as *fonts*. A font typically contains images that share a common appearance and look nice

together (for example, a single size, boldness, slant, and character set). Similarly, collections of fonts that are based on a common type face (the variations are usually called roman, bold, italic, bold italic, oblique, and bold oblique) are called *families*.

Sets of font families of the same resolution (usually measured in dots per inch) are further grouped into *directories* (so named because they were initially stored in file system directories). Each directory contains a database which lists the name of the font and information on how to find the font. The server uses these databases to translate *font names* (which have nothing to do with file names) into font data.

The list of font directories in which the server looks when trying to find a font is controlled by the *font path*. Although most installations will choose to have the server start up with all of the commonly used font directories, the font path can be changed at any time with the *xset* program. However, it is important to remember that the directory names are on the server's machine, not on the application's.

The default font path for the sample server contains three directories:

/usr/lib/X11/fonts/misc

This directory contains several miscellaneous fonts that are useful on all systems. It contains a very small family of fixed-width fonts (**6x10**, **6x12**, **6x13**, **8x13**, **8x13bold**, and **9x15**) and the cursor font. It also has font name aliases for the commonly used fonts *fixed* and *variable*.

/usr/lib/X11/fonts/75dpi

This directory contains fonts contributed by Adobe Systems, Inc. and Digital Equipment Corporation and by Bitstream, Inc. for 75 dots per inch displays. An integrated selection of sizes, styles, and weights are provided for each family.

/usr/lib/X11/fonts/100dpi

This directory contains 100 dots per inch versions of some of the fonts in the *75dpi* directory.

Font databases are created by running the *mkfontdir* program in the directory containing the source or compiled versions of the fonts (in both compressed and uncompressed formats). Whenever fonts are added to a directory, *mkfontdir* should be rerun so that the server can find the new fonts. To make the server reread the font database, reset the font path with the *xset* program. For example, to add a font to a private directory, the following commands could be used:

```
% cp newfont.snf ~/myfonts
% mkfontdir ~/myfonts
% xset fp rehash
```

The *xlsfonts* program can be used to list all of the fonts that are found in font databases in the current font path. Font names tend to be fairly long as they contain all of the information needed to uniquely identify individual fonts. However, the sample server supports wildcarding of font names, so the full specification

-adobe-courier-medium-r-normal--10-100-75-75-m-60-iso8859-1

could be abbreviated as:

```
*-courier-medium-r-normal--*-100-*
```

Because the shell also has special meanings for `*` and `?`, wildcarded font names should be quoted:

```
% xlsfonts -fn '*-courier-medium-r-normal--*-100-*
```

If more than one font in a given directory in the font path matches a wildcarded font name, the choice of which particular font to return is left to the server. However, if fonts from more than one directory match a name, the returned font will always be from the first such directory in the font path. The example given above will match fonts in both the *75dpi* and *100dpi* directories; if the *75dpi* directory is ahead of the *100dpi* directory in the font path, the smaller version of the font will be used.

COLOR NAMES

Most applications provide ways of tailoring (usually through resources or command line arguments) the colors of various elements in the text and graphics they display. Although black and white displays don't provide much of a choice, color displays frequently allow anywhere between 16 and 16 million different colors.

Colors are usually specified by their commonly-used names (for example, *red*, *white*, or *medium slate blue*). The server translates these names into appropriate screen colors using a color database that can usually be found in */usr/lib/X11/rgb.txt*. Color names are case-insensitive, meaning that *red*, *Red*, and *RED* all refer to the same color.

Many applications also accept color specifications of the following form:

```
#rgb
#rrggb
#rrrggbb
#rrrrgggbbbb
```

where *r*, *g*, and *b* are hexadecimal numbers indicating how much *red*, *green*, and *blue* should be displayed (zero being none and *ffff* being on full). Each field in the specification must have the same number of digits (e.g., *#rrgb* or *#gbb* are not allowed). Fields that have fewer than four digits (e.g. *#rgb*) are padded out with zero's following each digit (e.g. *#r000g000b000*). The eight primary colors can be represented as:

black	#000000000000 (no color at all)
red	#ffffff000000
green	#0000ffff0000
blue	#000000ffff
yellow	#ffffff0000 (full red and green, no blue)
magenta	#ffff0000ffff
cyan	#0000ffff00ffff
white	#ffffff00ffff (full red, green, and blue)

Unfortunately, RGB color specifications are highly unportable since different monitors produce different shades when given the same inputs. Similarly, color names aren't portable because there is no standard naming scheme and because the color database needs to be tuned for each monitor.

Application developers should take care to make their colors tailorable.

KEYS

The X keyboard model is broken into two layers: server-specific codes (called *keycodes*) which represent the physical keys, and server-independent symbols (called *keysyms*) which represent the letters or words that appear on the keys. Two tables are kept in the server for converting keycodes to keysyms:

modifier list

Some keys (such as Shift, Control, and Caps Lock) are known as *modifier* and are used to select different symbols that are attached to a single key (such as Shift-a generates a capital A, and Control-l generates a formfeed character ^L). The server keeps a list of keycodes corresponding to the various modifier keys. Whenever a key is pressed or released, the server generates an *event* that contains the keycode of the indicated key as well as a mask that specifies which of the modifier keys are currently pressed. Most servers set up this list to initially contain the various shift, control, and shift lock keys on the keyboard.

keymap table

Applications translate event keycodes and modifier masks into keysyms using a *keysym table* which contains one row for each keycode and one column for each of the modifiers. This table is initialized by the server to correspond to normal typewriter conventions, but is only used by client programs.

Although most programs deal with keysyms directly (such as those written with the X Toolkit), most programming libraries provide routines for converting keysyms into the appropriate type of string (such as ISO Latin-1). However, programs that use such routines are usually less portable and not as flexible.

OPTIONS

Most X programs attempt to use the same names for command line options and arguments. All applications written with the X Toolkit automatically accept the following options:

-display *display*

This option specifies the name of the X server to use.

-geometry *geometry*

This option specifies the initial size and location of the window.

-bg color, **-background** *color*

Either option specifies the color to use for the window background.

-bd color, **-bordercolor** *color*

Either option specifies the color to use for the window border.

-bw *number*, -borderwidth *number*

Either option specifies the width in pixels of the window border.

-fg *color*, -foreground *color*

Either option specifies the color to use for text or graphics.

-fn *font*, -font *font*

Either option specifies the font to use for displaying text.

-iconic

This option indicates that the user would prefer that the application's windows initially not be visible as if the windows had been immediately iconified by the user. Window managers may choose not to honor the application's request.

-name

This option specifies the name under which resources for the application should be found. This option is useful in shell aliases to distinguish between invocations of an application, without resorting to creating links to alter the executable file name.

-rv, -reverse

Either option indicates that the program should simulate reverse video if possible, often by swapping the foreground and background colors. Not all programs honor this or implement it correctly. It is usually only used on monochrome displays.

+rv

This option indicates that the program should not simulate reverse video. This is used to override any defaults since reverse video doesn't always work properly.

-synchronous

This option indicates that requests to the X server should be sent synchronously, instead of asynchronously. Since *Xlib* normally buffers requests to the server, errors do not necessarily get reported immediately after they occur. This option turns off the buffering so that the application can be debugged. It should never be used with a working program.

-title *string*

This option specifies the title to be used for this window. This information is sometimes used by a window manager to provide some sort of header identifying the window.

-xrm *resourcestring*

This option specifies a resource name and value to override any defaults. It is also very useful for setting resources that don't have explicit command line arguments.

RESOURCES

To make the tailoring of applications to personal preferences easier, X supports several mechanisms for storing default values for program resources (e.g. background color, window title, etc.) Resources are specified as strings of the form

*name*subname*subsubname...: value*

that are read in from various places when an application is run. The *Xlib* routine *XGetDefault(3X)* and the resource utilities within the X Toolkit obtain resources from the following sources:

RESOURCE_MANAGER root window property

Any global resources that should be available to clients on all machines should be stored in the RESOURCE_MANAGER property on the root window using the *xrdb* program. This is frequently taken care of when the user starts up X through the display manager or *xinit*.

application-specific files

Any application- or machine-specific resources can be stored in the class resource files located in the XAPPOADDIR directory (this is a configuration parameter that is */usr/lib/X11/app-defaults* in the standard distribution). Programs that use the X Toolkit will also look in the directory named by the environment variable XAPPLRESDIR (default value is user's home directory) for files named *Class* where *Class* is the class name of the particular application. XAPPOADDIR and XAPPLRESDIR configuration files are actually loaded *before* the RESOURCE_MANAGER property, so that the property can override the values.

XENVIRONMENT

Any user- and machine-specific resources may be specified by setting the XENVIRONMENT environment variable to the name of a resource file to be loaded by all applications. If this variable is not defined, the X Toolkit looks for a file named *.Xdefaults-hostname*, where *hostname* is the name of the host where the application is executing.

-xrm resourcestring

Applications that use the X Toolkit can have resources specified from the command line. The *resourcestring* is a single resource name and value as shown above. Note that if the string contains characters interpreted by the shell (e.g., asterisk), they must be quoted. Any number of **-xrm** arguments may be given on the command line.

Program resources are organized into groups called *classes*, so that collections of individual resources (each of which are called *instances*) can be set all at once. By convention, the instance name of a resource begins with a lowercase letter and class name with an upper case letter. Multiple word resources are concatenated with the first letter of the succeeding words capitalized. Applications written with the X Toolkit will have at least the following resources:

background (class Background)

This resource specifies the color to use for the window background.

borderWidth (class BorderWidth)

This resource specifies the width in pixels of the window border.

borderColor (class BorderColor)

This resource specifies the color to use for the window border.

Most X Toolkit applications also have the resource **foreground** (class **Foreground**), specifying the color to use for text and graphics within the window.

By combining class and instance specifications, application preferences can be set quickly and easily. Users of color displays will frequently want to set Background and Foreground classes to particular defaults. Specific color instances such as text cursors can then be overridden without having to define all of the related resources. For example,

```

bitmap*Dashed: off
XTerm*cursorColor: gold
XTerm*multiScroll: on
XTerm*jumpScroll: on
XTerm*reverseWrap: on
XTerm*curses: on
XTerm*Font: 6x10
XTerm*scrollBar: on
XTerm*scrollbar*thickness: 5
XTerm*multiClickTime: 500
XTerm*charClass: 33:48,37:48,45-47:48,64:48
XTerm*cutNewline: off
XTerm*cutToBeginningOfLine: off
XTerm*titeInhibit: on
XTerm*ttyModes: intr ^c erase ^? kill ^u
XLoad*Background: gold
XLoad*Foreground: red
XLoad*highlight: black
XLoad*borderWidth: 0
emacs*Geometry: 80x65-0-0
emacs*Background: #5b7686
emacs*Foreground: white
emacs*Cursor: white
emacs*BorderColor: white
emacs*Font: 6x10
xmag*geometry: -0-0
xmag*borderColor: white
uwm*bordercolor: black
uwm*Foreground: #f00
uwm*Background: #ede7e2
uwm*borderwidth: 0
uwm*iborderwidth: 0
uwm*mborderwidth: 1

```

If these resources were stored in a file called *.Xresources* in your home directory, they could be added to any existing resources in the server with the following command:

```
% xrb -merge $HOME/.Xresources
```

This is frequently how user-friendly startup scripts merge user-specific defaults into any site-wide defaults. All sites are encouraged to set up convenient ways of automatically loading resources. See the *Xlib* manual section *Using the Resource Manager* for more information.

EXAMPLES

The following is a collection of sample command lines for some of the more frequently used commands. For more information on a particular command, please refer to that command's manual page.

```
% xrb -load $HOME/.Xresources
% xmodmap -e "keysym BackSpace = Delete"
% mkfontdir /usr/local/lib/X11/otherfonts
% xset fp+ /usr/local/lib/X11/otherfonts
% xmodmap $HOME/.keymap.km
% xsetroot -solid '#888'
% xset b 100 400 c 50 s 1800 r on
% xset q
% uwm
% xmag
% xclock -geometry 48x48-0+0 -bg blue -fg white
% xeyes -geometry 48x48-48+0
% xbiff -update 20
% xlsfonts '*helvetica*'
% xlswins -l
% xwininfo -root
% xdpinfo -display joesworkstation:0
% xhost -joesworkstation
% xrefresh
% xwd | xwud
% bitmap companylogo.bm 32x32
% xcalc -bg blue -fg magenta
% xterm -geometry 80x66-0-0 -name myxterm $*
```

DIAGNOSTICS

A wide variety of error messages are generated from various programs. Various toolkits are encouraged to provide a common mechanism for locating error text so that applications can be tailored easily. Programs written to interface directly to the *Xlib* C language library are expected to do their own error checking.

The default error handler in *Xlib* (also used by many toolkits) uses standard resources to construct diagnostic messages when errors occur. The defaults for these messages are usually stored in */usr/lib/X11/XErrorDB*. If this file is not present, error messages will be rather terse and cryptic.

When the X Toolkit encounters errors converting resource strings to the appropriate internal format, no error messages are usually printed. This is convenient when it is desirable to have one set of resources across a variety of displays (e.g. color vs. monochrome, lots of fonts vs. very few, etc.), although it can pose problems for trying to determine why an application might be failing. This behavior can be overridden by the setting the *StringConversionsWarning* resource.

To force the Toolkit to always print string conversion error messages, the following resource should be placed at the top of the file that gets loaded onto the *RESOURCE_MANAGER* property using the *xrdb* program (frequently called *.Xresources* or *.Xres* in the user's home directory):

```
*StringConversionWarnings: on
```

To have conversion messages printed for just a particular application, the appropriate instance name can be placed before the asterisk:

```
xterm*StringConversionWarnings: on
```

BUGS

If you encounter a **repeatable** bug, please contact your site administrator for instructions on how to submit an X Bug Report.

SEE ALSO

Xserver(1), mkfontdir(1), bitmap(1), uwm(1), x10toX11(1), xbiff(1), xcalc(1), xclock(1), xdpinfo(1), xedit(1), xev(1), xfd(1), xhost(1), xinit(1), xkill(1), xload(1), xlogo(1), xlsfonts(1), xlswins(1), xmag(1), xman(1), xmh(1), xmodmap(1), xpr(1), xprop(1), xrdb(1), xrefresh(1), xset(1), xsetroot(1), resize(1), xterm(1), xterm(7), xwd(1), xwininfo(1), xwud(1), Xapollo(1), bdf2otf(1), /com/help kbd, biff(1), init(8), ttys(5), *Xlib - C Language X Interface*, *X Toolkit Intrinsics - C Language X Interface*, and *Using and Specifying X Resources*.

COPYRIGHT

The following copyright and permission notice outlines the rights and restrictions covering most parts of the standard distribution of the X Window System from MIT. Other parts have additional or different copyrights and permissions; see the individual source files.

Copyright 1984, 1985, 1986, 1987, 1988, Massachusetts Institute of Technology. Apollo-specific changes Copyright 1989 by Apollo Computer Inc.

Permission to use, copy, modify, and distribute this software and its documentation for any purpose and without fee is hereby granted, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation, and that the name of M.I.T. not be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission. M.I.T. makes no representations about the suitability of this software for any purpose. It is provided "as is" without express or implied warranty.

This software is not subject to any license of the American Telephone and Telegraph Company or of the Regents of the University of California.

AUTHORS

A cast of thousands. See the file *doc/contributors* in the standard sources for some of the names.

NAME

Xapollo – Apollo Domain/X11 "shared" X window system server

SYNOPSIS

Xapollo [-D1] [s{+|-}r{+|-}] [-K *keyboardfile*]

DESCRIPTION

Xapollo is the Apollo Domain/X11 "shared" X window system server. It supports the MIT X11 version of the X protocol.

Xapollo is called a "shared server" because it can run in two modes wherein X and Apollo Display Manager (DM) windows are displayed on the screen together, "sharing" the display.

See below under "Implementation Notes" for a detailed description of server features.

CONFIGURATIONS

Xapollo runs on Domain/OS, under Apollo software release SR10.2. TCP/IP, available at SR10 as a bundled product, *must* also be installed and running to take full advantage of X's remote display ability, but it is not necessary for local-only connections using Unix domain sockets.

Xapollo operates on any Apollo color or monochrome display hardware. It requires a keyboard with a mouse.

OPTIONS

The *Xapollo* server program takes the following device dependent options. In addition, the "standard" X11 server command line options can also be used. These options are described under *Xserver(1)*.

-D<unit>

This option gives the display unit number to which the following **s** (share mode) and **r** (own root) options should apply. Since at present Apollo does not support multiple display devices, only **-D1**, meaning display unit 1, is meaningful. **-D1** is the default.

s{+|-}r{+|-}

s specifies whether the Domain/X11 server should "share" the display with the Apollo Display Manager (DM). **s+** means, "yes, share the display." In this mode, you can have both DM and X clients on the screen at the same time. **s+** is the default.

s- means "do not share the display." In this mode, the Domain/X11 server will borrow the entire display and you can run only X clients. (This is the way that the public-domain servers from MIT and from ADUS behave.) If you are going to use **s-**, you should give the *Xapollo* command as part of an *xinit(1)* command so that you can also start an *xterm(1)* at the same time. This way, when the server borrows the display you have something you can type to. See the man page for *xinit(1)*.

r{+|-} specifies (given **s+**) which of the two Apollo window systems--the DM or X--should own the root (background) window and be responsible for

window management. **r-** means that the DM should own the root. It is the default. **r+** means that Domain/X11 should own the root window.

s and **r** should form a single string without embedded blanks, e.g.: **s+r-**

-K *keyboardfile*

The **-K** option specifies a keyboard configuration file. **-K** is useful when:

- The server is being run **s+r-** (share mode, DM owning the root). You need the configuration file when the DM owns the root so that you can Pop, Move, and Grow X windows.
- The server is being run in X-owns root or borrow mode. In this case the configuration file can specify a key to use as a "Quit server" key.

If **-K** *keyboardfile* is not specified, by default all keys go to the X server. By default, there is no "Quit server" key.

The keyboard configuration file contains simple one-statement-per-line commands that parcel out the keys on the keyboard (either keyboard 2 or 3) between the DM and X. There are three commands:

quit *<Xkeysym>*

Names a key that will exit the X server when pressed in "X territory." X territory is any X window listening to keypress events, or the background root window if X owns the root. (Most X clients take keypress events e.g., *xterm(1)*. A very few-- *xclock(1)* is one--do not.)

The main purpose here is to provide a way to kill the X server and return the DM if the X server should hang in borrow mode. The quit key will work in any server mode, however. By default, there is no quit key.

exclude *<Xkeysym>*

When the X server initializes, by default it "takes" all keys on the keyboard to itself. If you press a key in X territory, then X gets first crack at the keyboard event. (Only in the rare "no keypress event" clients like *xclock* does the event pass through to the parent window--which might be the background owned by X or the DM.)

This behavior is in keeping with the idea that X should be able to run on Apollos as though it were the only window system present, and clients should be able to assume they can use the whole keyboard.

Practically, though, there are keys you want the DM to receive, even if they are pressed in X territory. For example, if the DM owns the root and is handling window management, you probably want Pop, Move/Grow, Cmd, NextWdw, etc. to work in X windows.

Saying "exclude *<Xkeysym>*" means "exclude this key from X and give it to the DM."

! Comment character. (After the fashion of *xmodmap(1)*.) This must appear in column 1 in order to denote that the line is a comment.

/usr/lib/X11/keyboard/keyboard.config is a sample keyboard configuration file. All non-white-key X keysyms on Apollo keyboard 2 (no keypad) and keyboard 3 (keypad and F0, F9) are shown. You should copy this to your home directory and customize it as you see fit. A reasonable choice of DM keys are already listed as "excluded" meaning that they will go to the DM no matter what territory they are pressed in. A Quit key command is commented out.

STARTING XAPOLLO

We recommend that you start Xapollo at boot time with the configuration files specified for it in */etc/rc*, by adding an empty file named Xapollo in your */etc/daemons* directory:

```
touch /etc/daemons/Xapollo
```

You may also want to start Xapollo at login by adding the line

```
/usr/bin/X11/startx
```

to one of your startup files (e.g., *.login*). The *startx* script uses *xinit* to start the server. In addition, it references the files *xinitrc* and *xserverrc* in */usr/lib/X11/xinit*. (We have placed other files in that directory that you may wish to copy into your home directory as files *.xinitrc* and *.xserverrc*, depending on whether you want to start Xapollo in DM- or X-root modes.) See */usr/lib/X11/xinit/xserverrc* for the *exec* command that starts the server.

Do not run the server from the command line using only Xapollo <options> &

IMPLEMENTATION NOTES

What Is a Shared Server?

Xapollo is called a "shared server" because X and DM windows can share the screen in two modes:

- The DM "owns" the background (root) window, draws the cursor as its familiar arrow, and does all window management (move, resize, iconify, etc.) for *both* X and DM windows. The DM acts as the primary window system.
- The X window System "owns" the background (root) window and draws the cursor as its familiar "X". DM window management has been explicitly turned off. An X window manager (probably *uwm(1)*) has been started and is used to move, resize, iconify, etc., both X and DM windows. X is the primary window system.

"Shared" can be contrasted with "borrow mode." The ADUS Apollo X11 and MIT-tape X11 Apollo servers were both "borrow mode" servers where the X server takes over the entire display. You can switch back and forth between X and the DM, but you cannot display both types of windows at once.

Having the Domain/X11 shared server makes it possible, for example, to use the Apollo *debug* utility (which is written to use DM PAD calls) to debug an X client and view both on the screen simultaneously.

DM PAD call and/or Apollo native graphics-based programs do not have to be recoded, recompiled, or relinked in order to work with the Domain/X11 server in either share mode. If it did run, it will run.

X clients linked to the ADUS X11 Apollo *Xlib* should be relinked with the Domain/X11 *Xlib*.

"Shared server" does not mean that one program has been written that acts as both an X and a DM window system. The *Xapollo* server and the DM remain distinct programs.

X and the DM are able to co-exist because of a pair of low-level Apollo system facilities called the Rectangle Manager (RM) and Input Demultiplexor (IDM). Introduced at SR9.5, these internal facilities underlie both X and the DM.

The Rectangle Manager (RM) manages a workstation global database of rectangular display areas on the screen. The database describes size, location, the hierarchical stacking order of the rectangles, parent/sibling relationships, which windows are obscured, and which process owns them. RM does not "draw" anything on the screen; that is the responsibility of the owning process. It is just a database of rectangle trees.

The Input Demultiplexor (IDM) controls workstation input from the keyboard and mouse. It maintains a global database of hierarchical input focus trees (related to the RM's rectangles), event queues, and provides a mechanism to bind particular devices, events, and processes together.

The RM/IDM together form a kind of window system in the abstract. The RM keeping an abstract representation of the screen layout, the IDM an abstract model of keyboard and mouse input.

Both the Domain/X11 X server and the DM use RM calls to create and manipulate windows. Both use IDM calls to handle input events. Because there is one database with window and input information that both appeal to, X and the DM can manipulate each others' windows and input is routed properly. So subtle a problem as passing the keyboard focus back and forth between X and the DM as the cursor slides over a random mixture of X and DM windows on the screen can be handled smoothly.

Normally, RM and IDM are invisible to the user, but it occasionally helps to know that this is what is actually going on. RM/IDM are part of */lib/gprlib*.

Native Graphics Protocol Extension

Apollo has added one extension to the X11 protocol. This extension lets X clients initialize and call Apollo native graphics routines in a subwindow. It is therefore possible to write a program that has an X interface (whether straight *Xlib*, Open Dialogue, the MIT X Toolkit, etc.) surrounding, say, a 3D GMR, GPR, or PHIGS graphics application.

Such hybrid clients will not run across a remote client/server connection (because Apollo native graphics are not protocol requests) nor are they directly portable to non-Apollo hardware platforms. They do, however give access to the speed of native graphics, provide a halfway house for vendors trying to port existing native graphics programs to X, and provide a way to do 3-D graphics, which X presently lacks.

This mechanism has come to be called "disowning a window." Example programs are in */usr/X11/examples/x_and_gpr* and */usr/X11/examples/x_and_3dgmw*.

Server Features

- The server supports monochrome, 4, or 8-plane color. The number of planes is detected automatically at server startup.
- When running on monochrome DN300, DN330, DN440, DN460 nodes, the server uses the portable but slow Monochrome Frame Buffer (MFB) driver code. On all other nodes, whether color or monochrome (DN560, DN570, DN580, DN590, DN3000, DN3500, DN4000) the Domain/X11 server is optimized to use Apollo native graphics interfaces.
- The server calls the Apollo Color Table Manager (CTM) to allocate/free color cells in the color map. It will thus not interfere with other programs that are well-behaved enough to use the CTM.
- The server provides full support, as defined in the X Protocol, for Apollo keyboard 3. (Keyboard 2 does not have hardware "up" transitions, such transitions are emulated.) Apollo international keyboards 3a-g are also supported, meaning that appropriate keysyms are assigned from the Latin 1 alphabet to the different multinational boards automatically at server initialization.

The exception to keyboard support is ChangeKeyboardControl (controlled with the *xset(1)* command). *xset(1)* functions controlling the pointer motion buffer, threshold and acceleration factors are not implemented. Apollo keyboard hardware does not allow for implementing *xset(1)*'s keyclick, bell pitch, or volume control.

- Domain/X11 includes a set of FORTRAN and Pascal "interludes" that make it possible to write Pascal or FORTRAN clients that call *Xlib*, which is written in C. The interludes bridge the incompatible conventions of case and parameter-passing between C and Pascal and Fortran. Interludes exist only for *Xlib* subroutine calls; the many macros do not have interludes.
- The Domain/X11 libraries have also been slightly modified so that they can be linked to X clients that were compiled and type-stamped under the "bsd4.2," "bsd4.3," or "sys5.3" environments. *Xlib* and the Toolkit library are now run-time libraries with globally-known symbols and must be included in the client node's */etc/sys.conf* files with the lines

```
lib x11lib, optional
lib xtlb, optional
```

In addition, the following libraries must be inlib'ed:

```
/lib/xawlib  
/lib/x11paslib  
/usr/X11/lib/libXmu.a
```

ERROR LOGGING

Domain/X11 server errors are logged in the file `'node_data/system_logs/X0msgs'`, which is a link to `/usr/adm/X0msgs` of errors. If possible, it includes fault diagnostic information and a traceback.

LIMITATIONS

You cannot cut/paste between X and DM windows. X and DM windows do not share X window properties.

The underlying support necessary to make "reparenting" window managers such as *twm(1)* work on both X and DM windows is not fully implemented. You can run *twm(1)* and it does work, but anomalous behavior does occur, particularly surrounding newly-created DM windows.

SEE ALSO

X(1), Xserver(1), xinit(1), xownroot(1).

COPYRIGHT

Copyright 1988, 1989, Apollo Computer Inc.

AUTHORS

Joe Bowbeer, Dave Gorgen, Jim Hamilton, Steve Reber, Bob Terek, Craig Wolpert
Apollo Computer Inc.

NAME

xdmc – execute a DM command from the shell

SYNOPSIS

xdmc *dm_command* [*args...*]

DESCRIPTION

xdmc allows you to invoke Display Manager commands from the command shell or from within a shell script. This is similar to pressing <CMD> on the keyboard and then typing the DM command in the DM input window, which is the usual way to perform DM operations.

dm_command (required)

Specifies the Display Manager command to be executed.

args ... (optional)

Specifies any arguments to be passed to the DM command. These are sent directly to the DM without further processing by the command shell.

Default if omitted: no arguments passed

EXAMPLES

\$ **xdmc** dq

Cause the DM to send a quit fault to the current process.

\$ **xdmc** cp /com/sh

Cause the DM to create a new process and invoke the shell. This is the same as pressing <SHELL>.

NAME

xownroot – switch between X and DM ownership of the root while the X server is running

SYNOPSIS

xownroot [-help] [-display *display*] [-on] [-off]

DESCRIPTION

The *xownroot* program lets you transfer ownership of the background root window between the shared X server and the Apollo DM. You can perform the switch between the two environments (between *Xapollo(1)*'s *s+r+* and *s+r-* modes) while the shared X server is running without needing to kill and restart it.

xownroot calls *XapolloGetRectangleForWindow* to discover the Rectangle Manager rectangle id for the root window. It then either "disowns" the root window with *XapolloDisownWindow* or takes responsibility for it by "owning" it with *XapolloOwnRectangle*. *xownroot* is thus neither taking the root away from anything specific, nor giving it to anything specific. You must manually turn on/off X window management by starting/killing an X window manager such as *uwm*. Similarly, you must explicitly tell the DM to assume/give up responsibility for the root window and turn on/off its window management functions with the *wmgr -on/off* commands. Indeed, the system taking responsibility for the root and window management need not necessarily be the DM.

Trying to run both systems' window managers at once is likely to lead to abnormal behavior and possible hangs. Problems with displaying the root background or other abnormalities might occur if both systems own the root window. If X owns the root, use an X window manager; if DM owns the root, use the DM as the window manager.

xownroot is thus expected to exist within a shell script such as:

tox:

```
#!/bin/csh
/com/xdmc wmgr -off
xownroot -on
uwm &
```

todm

```
#!/bin/csh
set wmpid='/bin/ps ax|grep "*wm"|head -1|/bin/awk '{print $1}'"
while ( $wmpid != "" )
  kill $wmpid; sleep 3
  set wmpid='/bin/ps ax|grep "*wm"|head -1|/bin/awk '{print $1}'"
end
usr/X11/bin/xownroot -off
usr/apollo/bin/xdmc wmgr -on
```

If no options are specified, the current mode is toggled.

OPTIONS

The various options are as follows:

- help** Print a usage message and exit.
- on** Own the root window.
- off** Disown the root window.
- display** *display*
 Specifies the server to connect to; see *X(1)*.

SEE ALSO

X(1), *Xapollo(1)*

COPYRIGHT

Copyright 1988, 1989, Apollo Computer, Inc.
See *X(1)* for a full statement of rights and permissions.

AUTHOR

Joe Bowbeer, Apollo Computer, Inc.

NAME

X - X Window System server

SYNOPSIS

X [:displaynumber] [-option ...] [ttyname]

DESCRIPTION

X is the generic name for the X Window System server. It is frequently a link or a copy of the appropriate server binary for driving the most frequently used server on a given machine. On Apollo systems, Xapollo is the name of the X server.

STARTING THE SERVER

Xapollo is typically started by the user at login via the *xinit* client or at workstation boot time using */etc/rc.local*, but on Apollo systems we recommend starting it in */etc/rc* as a daemon or via the *startx* script. See Xapollo(1).

On non-Apollo systems, the server is usually started from the X Display Manager program *xdm*. This utility is run from the system boot files and takes care of keeping the server running, prompting for usernames and passwords, and starting up the user sessions. It is easily configured for sites that wish to provide nice, consistent interfaces for novice users (loading convenient sets of resources, starting up a window manager, clock, and nice selection of terminal emulator windows).

Since *xdm* now handles automatic starting of the server in a portable way, the *-L* option to *xterm* is now considered obsolete. Support for starting a login window from 4.3bsd-derived */etc/tty*s files may not be included in future releases.

Installations that run more than one window system will still need to use the *xinit* utility. However, *xinit* is to be considered a tool for building startup scripts and is not intended for use by end users.

NETWORK CONNECTIONS

The server supports connections made using the following reliable byte-streams:

TCP/IP

The server listens on port *htons(6000+n)*, where *n* is the display number.

Unix Domain

The server uses */tmp/X11-unix/Xn* as the filename for the socket, where *n* is the display number.

DECnet

The server responds to connections to object *X\$Xn*, where *n* is the display number.

OPTIONS

The following command line options are accepted:

-a number

sets pointer acceleration (i.e. the ratio of how much is reported to how much the user actually moved the pointer).

-bs

disables backing store support on all screens. [Not available on Xapollo.]

- c** turns off key-click.
- c volume** sets key-click volume (allowable range: 0-8).
- f volume** sets feep (bell) volume (allowable range: 0-7).
- logo** turns on the X Window System logo display in the screen-saver. There is currently no way to change this from a client.
- nologo** turns off the X Window System logo display in the screen-saver. There is currently no way to change this from a client.
- p minutes** sets screen-saver pattern cycle time in minutes.
- r** turns off auto-repeat.
- r** turns on auto-repeat.
- s minutes** sets screen-saver timeout time in minutes.
- su** disables save under support on all screens. [Not available on Xapollo.]
- t numbers** sets pointer acceleration threshold in pixels (i.e. after how many pixels pointer acceleration should take effect).
- to seconds** sets default screensaver timeout in seconds.
- v** sets video-on screen-saver preference.
- v** sets video-off screen-saver preference
- co filename** sets name of RGB color database
- help** prints a usage message
- fp fontPath** sets the search path for fonts
- fc cursorFont** sets default cursor font
- fn font** sets the default font
- wm** forces the default backing-store of all windows to be `WhenMapped`; a cheap trick way of getting backing-store to apply to all windows. [Not available on Xapollo.] See the manual pages for Xapollo for more Apollo-specific details.

SECURITY

The server uses an access control list for deciding whether or not to accept connections from clients on a particular machine. This list initially consists of the host on which the server is running as well as any machines listed in the file `/etc/Xn.hosts`, where `n` is the display number of the server. Each line of the file should contain

either an Internet hostname (e.g. expo.lcs.mit.edu) or a DECnet hostname in double colon format (e.g. hydra::). There should be no leading or trailing spaces on any lines. For example:

```
joesworkstation
corporate.company.com
star::
bigcpu::
```

Users can add or remove hosts from this list and enable or disable access control using the *xhost* command from the same machine as the server. For example:

```
% xhost +janesworkstation
janesworkstation being added to access control list
% xhost -star::
public:: being removed from access control list
% xhost +
all hosts being allowed (access control disabled)
% xhost -
all hosts being restricted (access control enabled)
% xhost
access control enabled (only the following hosts are allowed)
joesworkstation
janesworkstation
corporate.company.com
bigcpu::
```

Unlike some window systems, X does not have any notion of window operation permissions or place any restrictions on what a client can do; if a program can connect to a display, it has full run of the screen. The core protocol does have support for providing authentication information when establishing connections, but is not used in the sample implementation. Sites that have authentication and authorization systems (such as Kerberos) might wish to make use of the hooks in the libraries and the server to provide additional security.

SIGNALS

The server attaches special meaning to the following signals:

SIGHUP This signal causes the server to close all existing connections, free all resources, and restore all defaults. It is sent by the display manager whenever the main user's main application (usually an *xterm* or window manager) exits to force the server to clean up and prepare for the next user.

SIGTERM

This signal causes the server to exit cleanly.

FONTS

Fonts are usually stored as individual files in directories. The list of directories in which the server looks when trying to open a font is controlled by the *font path*. Although most sites will choose to have the server start up with the appropriate font

path (using the *-fp* option mentioned above), it can be overridden using the *xset* program.

The default font path for the sample server contains three directories:

/usr/lib/X11/fonts/misc

This directory contains several miscellaneous fonts that are useful on all systems. It contains a very small family of fixed-width fonts (**6x10**, **6x12**, **6x13**, **8x13**, **8x13bold**, and **9x15**) and the cursor font. It also has font name aliases for the commonly used fonts **fixed** and **variable**.

/usr/lib/X11/fonts/75dpi

This directory contains fonts contributed by Adobe Systems, Inc. and Digital Equipment Corporation and by Bitstream, Inc. for 75 dots per inch displays. An integrated selection of sizes, styles, and weights are provided for each family.

/usr/lib/X11/fonts/100dpi

This directory contains versions of some of the fonts in the *75dpi* directory for 100 dots per inch displays.

Font databases are created by running the *mkfontdir* program in the directory containing the compiled versions of the fonts (the *.snf* files). Whenever fonts are added to a directory, *mkfontdir* should be rerun so that the server can find the new fonts. **If *mkfontdir* is not run, the server will not be able to find any fonts in the directory.**

DIAGNOSTICS

Too numerous to list them all. If run from *init(8)*,

FILES

<i>/etc/X*.hosts</i>	Initial access control list
<i>/usr/lib/X11/fonts/misc</i> , <i>/usr/lib/X11/fonts/75dpi</i> , <i>/usr/lib/X11/fonts/100dpi</i>	Font directories
<i>/usr/lib/X11/rgb.txt</i>	Color database
<i>/tmp/.X11-unix/X*</i>	Unix domain socket
<i>/usr/adm/X*msgs</i>	Error log file

SEE ALSO

X(1), *xdm(1)*, *mkfontdir(1)*, *xinit(1)*, *xterm(1)*, *uwm(1)*, *xhost(1)*, *xset(1)*, *xsetroot(1)*, *ttys(5)*, *init(8)*, *Xapollo(1)*, *X Window System Protocol, Definition of the Porting Layer for the X v11 Sample Server*, *Strategies for Porting the X v11 Sample Server*, *Godzilla's Guide to Porting the X V11 Sample Server*

BUGS

The option syntax is inconsistent with itself and *xset(1)*.

The acceleration option should take a numerator and a denominator like the protocol.

If *X* dies before its clients, new clients won't be able to connect until all existing connections have their TCP TIME_WAIT timers expire.

The color database is missing a large number of colors. However, there doesn't seem to be a better one available that can generate RGB values tailorable to particular displays.

The *xterm -L* method for starting an initial window from */etc/tty*s is completely inadequate and should be removed. People should use *xdm* instead.

COPYRIGHT

Copyright 1984, 1985, 1986, 1987, 1988, Massachusetts Institute of Technology.
Apollo-specific changes Copyright 1989 Apollo Computer Inc.

See *X(1)* for a full statement of rights and permissions.

AUTHORS

The sample server was originally written by Susan Angebrandt, Raymond Drewry, Philip Karlton, and Todd Newman, with support from a cast of thousands. See also the file *doc/contributors* in the sample distribution for a more complete list.

NAME

xset - user preference utility for X

SYNOPSIS

```
xset [-display display] [-b] [b on/off] [b [volume [pitch [duration]]] [-c] [c on/off]
[c [volume]] [[-+]fp[-+=] path[,path[...]]] [fp default] [fp rehash] [[-]led [integer]]
[led on/off] [m[ouse] [acceleration [threshold]]] [m[ouse] default] [p pixel color]
[[-]r] [r on/off] [s [length [period]]] [s blank/noblink] [s expose/noexpose] [s
on/off] [s default] [q]
```

DESCRIPTION

This program is used to set various user preference options of the display.

OPTIONS

-display *display*

This option specifies the server to use; see *X(1)*.

- b** the **b** option controls bell volume, pitch and duration. This option accepts up to three numerical parameters, a preceding dash(-), or a 'on/off' flag. If no parameters are given, or the 'on' flag is used, the system defaults will be used. If the dash or 'off' are given, the bell will be turned off. If only one numerical parameter is given, the bell volume will be set to that value, as a percentage of its maximum. Likewise, the second numerical parameter specifies the bell pitch, in hertz, and the third numerical parameter specifies the duration in milliseconds. Note that not all hardware can vary the bell characteristics. The X server will set the characteristics of the bell as closely as it can to the user's specifications.
- c** The **c** option controls key click. This option can take an optional value, a preceding dash(-), or an 'on/off' flag. If no parameter or the 'on' flag is given, the system defaults will be used. If the dash or 'off' flag is used, key-click will be disabled. If a value from 0 to 100 is given, it is used to indicate volume, as a percentage of the maximum. The X server will set the volume to the nearest value that the hardware can support.

fp= *path*,...

The **fp=** sets the font path to the directories given in the path argument. The directories are interpreted by the server, not by the client, and are server-dependent. Directories that do not contain font databases created by *mkfontdir* will be ignored by the server.

fp *default*

The *default* argument causes the font path to be reset to the server's default.

fp *rehash*

The *rehash* argument causes the server to reread the font databases in the current font path. This is generally only used when adding new fonts to a font directory (after running *mkfontdir* to recreate the font database).

-fp or **fp-**

The **-fp** and **fp-** options remove elements from the current font path. They must be followed by a comma-separated list of directories.

+fp or fp+

This **+fp** and **fp+** options prepend and append elements to the current font path, respectively. They must be followed by a comma-separated list of directories.

led

The **led** option controls the keyboard LEDs. This controls the turning on or off of one or all of the LEDs. It accepts an optional integer, a preceding dash(-) or an 'on/off' flag. If no parameter or the 'on' flag is given, all LEDs are turned on. If a preceding dash or the flag 'off' is given, all LEDs are turned off. If a value between 1 and 32 is given, that LED will be turned on or off depending on the existence of a preceding dash. A common LED which can be controlled is the "Caps Lock" LED. "xset led 3" would turn led #3 on. "xset -led 3" would turn it off. The particular LED values may refer to different LEDs on different hardware.

m

The **m** option controls the mouse parameters. The parameters for the mouse are 'acceleration' and 'threshold'. The mouse, or whatever pointer the machine is connected to, will go 'acceleration' times as fast when it travels more than 'threshold' pixels in a short time. This way, the mouse can be used for precise alignment when it is moved slowly, yet it can be set to travel across the screen in a flick of the wrist when desired. One or both parameters for the **m** option can be omitted, but if only one is given, it will be interpreted as the acceleration. If no parameters or the flag 'default' is used, the system defaults will be set.

p

The **p** option controls pixel color values. The parameters are the color map entry number in decimal, and a color specification. The root background colors may be changed on some servers by altering the entries for BlackPixel and WhitePixel. Although these are often 0 and 1, they need not be. Also, a server may choose to allocate those colors privately, in which case an error will be generated. The map entry must not be a read-only color, or an error will result.

r

The **r** option controls the autorepeat. If a preceding dash or the 'off' flag is used, autorepeat will be disabled. If no parameters or the 'on' flag is used, autorepeat will be enabled.

s

The **s** option lets you set the screen saver parameters. This option accepts up to two numerical parameters, a 'blank/noblink' flag, an 'expose/noexpose' flag, an 'on/off' flag, or the 'default' flag. If no parameters or the 'default' flag is used, the system will be set to its default screen saver characteristics. The 'on/off' flags simply turn the screen saver functions on or off. The 'blank' flag sets the preference to blank the video (if the hardware can do so) rather than display a background pattern, while 'noblink' sets the preference to display a pattern rather than blank the video. The 'expose' flag sets the preference to allow window exposures (the server can freely discard window contents), while 'noexpose' sets the preference to disable screen saver unless the server can regenerate the screens without causing exposure events. The length and period parameters for the screen saver function determines how long the server must be

inactive for screen saving to activate, and the period to change the background pattern to avoid burn in. The arguments are specified in seconds. If only one numerical parameter is given, it will be used for the length.

q The **q** option gives you information on the current settings.

These settings will be reset to default values when you log out.

Note that not all X implementations are guaranteed to honor all of these options. [The Apollo SR10.2 implementation does not honor options **b**, **c**, **led**, and **m**, in addition to the 'noblack' and 'expose' flags on the **s** option. These options and flags can be set, but the setting is not implemented and no notice is given the user.]

SEE ALSO

X(1), Xserver(1), xmodmap(1), xrdp(1), xsetroot(1)

COPYRIGHT

Copyright 1988, Massachusetts Institute of Technology. Apollo implementation note
Copyright 1989, Apollo Computer Inc.

See *X(1)* for a full statement of rights and permissions.

AUTHOR

Bob Scheifler, MIT Laboratory for Computer Science
David Krikorian, MIT Project Athena (X11 version)

NAME

xstr — extract strings from C programs to implement shared strings

SYNOPSIS

xstr [**-c**] [**-**] [*file*]

DESCRIPTION

xstr maintains a file **strings** into which strings in component parts of a large program are hashed. **xstr** replaces these strings with references to this common area. This serves to implement shared constant strings, which are most useful if they are also read-only.

The command

xstr -c name

extracts the strings from the C source in *name*, replacing string references by expressions of the form (**&xstr[*number*]**) for some number. An appropriate declaration of **xstr** is prefixed to the file. The resulting C text is placed in the file **x.c** to be compiled. The strings from this file are placed in the **strings** database if they are not there already. Repeated strings and strings that are suffixes of existing strings do not cause changes to the database.

After all components of a large program have been compiled, a file **xs.c** declaring the common **xstr** space can be created by a command of the form

xstr

This **xs.c** file should then be compiled and loaded with the rest of the program. If possible, the array can be made read-only (shared) saving space and swap overhead.

xstr can also be used on a single file. A command

xstr name

creates files **x.c** and **xs.c** as before, without using or affecting any **strings** file in the same directory.

It may be useful to run **xstr** after the C preprocessor if any macro definitions yield strings or if there is conditional code that contains strings which may not, in fact, be needed. **xstr** reads from its standard input when the dash argument (**-**) is given. An appropriate command sequence for running **xstr** after the C preprocessor is

```
cc -E name.c | xstr -c -
cc -c x.c
mv x.o name.o
```

xstr does not touch the file **strings** unless new items are added; thus **make** can avoid remaking **xs.o** unless truly necessary.

FILES

strings	Database of strings
x.c	Massaged C source
xs.c	C source for definition of array 'xstr'

/tmp/xs* Temp file when 'xstr name' doesn't touch **strings**

BUGS

If a string is a suffix of another string in the database but the shorter string is seen first by **xstr**, both strings will be placed in the database when just placing the longer one there will do.

SEE ALSO

mkstr(1)

NAME

xterm – terminal emulator for X

SYNOPSIS

xterm [-*toolkitoption* ...] [-option ...]

DESCRIPTION

The *xterm* program is a terminal emulator for the X Window System. It provides DEC VT102 and Tektronix 4014 terminal emulation for programs that can't use the window system directly.

The VT102 and Tektronix 4014 terminals each have their own window so that you can edit text in one and look at graphics in the other at the same time. To maintain the correct aspect ratio (height/width), Tektronix graphics will be restricted to the largest box with a 4014's aspect ratio that will fit in the window. This box is located in the upper left area of the window.

Although both windows may be displayed at the same time, one of them is considered the "active" window for receiving keyboard input and terminal output. This is the window that contains the text cursor and whose border highlights whenever the pointer is in either window. The active window can be chosen through escape sequences, the "Modes" menu in the VT102 window, and the "Tektronix" menu in the 4014 window.

OPTIONS

The *xterm* terminal emulator accepts all of the standard X Toolkit command line options along with the additional options listed below (if the option begins with a '+' instead of a '-', the option is restored to its default value):

- help** This causes *xterm* to print out a verbose message describing its options.
- 132** Normally, the VT102 DECCOLM escape sequence that switches between 80 and 132 column mode is ignored. This option causes the DECCOLM escape sequence to be recognized, and the *xterm* window will resize appropriately.
- ah** This option indicates that *xterm* should always highlight the text cursor and borders. By default, *xterm* will display a hollow text cursor whenever the focus is lost or the pointer leaves the window.
- +ah** This option indicates that *xterm* should do text cursor highlighting.
- b *number***
This option specifies the size of the inner border (the distance between the outer edge of the characters and the window border) in pixels. The default is 2.
- cc *characterclassrange:value[,...]***
This sets classes indicated by the given ranges for using in selecting by words. See the section specifying character classes.

-cr color

This option specifies the color to use for text cursor. The default is to use the same foreground color that is used for text.

-cu

This option indicates that *xterm* should work around a bug in the *curses*(3x) cursor motion package that causes the *more*(1) program to display lines that are exactly the width of the window and are followed by a line beginning with a tab to be displayed incorrectly (the leading tabs are not displayed).

+cu

This option indicates that that *xterm* should not work around the *curses*(3x) bug mentioned above.

-e program [arguments ...]

This option specifies the program (and its command line arguments) to be run in the *xterm* window. It also sets the window title and icon name to be the basename of the program being executed if neither *-T* nor *-n* are given on the command line. **This must be the last option on the command line.**

-fb font

This option specifies a font to be used when displaying bold text. This font must be the same height and width as the normal font. If only one of the normal or bold fonts is specified, it will be used as the normal font and the bold font will be produced by overstriking this font. The default bold font is "vtbold."

-j

This option indicates that *xterm* should do jump scrolling. Normally, text is scrolled one line at a time; this option allows *xterm* to move multiple lines at a time so that it doesn't fall as far behind. Its use is strongly recommended since it make *xterm* much faster when scanning through large amounts of text. The VT100 escape sequences for enabling and disabling smooth scroll as well as the "Modes" menu can be used to turn this feature on or off.

+j

This option indicates that *xterm* should not do jump scrolling.

-l

This option indicates that *xterm* should send all terminal output to a log file as well as to the screen. This option can be enabled or disabled using the "xterm X11" menu.

+l

This option indicates that *xterm* should not do logging.

-lf filename

This option specifies the name of the file to which the output log described above is written. If *file* begins with a pipe symbol (*|*), the rest of the string is assumed to be a command to be used as the endpoint of a pipe. The default filename is "XtermLog.XXXXX" (where XXXXX is the process id of *xterm*) and is created in the directory from which *xterm* was started (or the user's home directory in the case of a login window).

-ls

This option indicates that the shell that is started in the *xterm* window be a login shell (i.e. the first character of *argv*[0] will be a dash, indicating to the shell that it should read the user's *.login* or *.profile*).

- +ls** This option indicates that the shell that is started should not be a login shell (i.e. it will be normal "subshell").
- mb** This option indicates that *xterm* should ring a margin bell when the user types near the right end of a line. This option can be turned on and off from the "Modes" menu.
- +mb** This option indicates that margin bell should not be rung.
- ms *color***
This option specifies the color to be used for the pointer cursor. The default is to use the foreground color.
- nb *number***
This option specifies the number of characters from the right end of a line at which the margin bell, if enabled, will ring. The default is 10.
- rw** This option indicates that reverse-wraparound should be allowed. This allows the cursor to back up from the leftmost column of one line to the rightmost column of the previous line. This is very useful for editing long shell command lines and is encouraged. This option can be turned on and off from the "Modes" menu.
- +rw** This option indicates that reverse-wraparound should not be allowed.
- s** This option indicates that *xterm* may scroll asynchronously, meaning that the screen does not have to be kept completely up to date while scrolling. This allows *xterm* to run faster when network latencies are very high and is typically useful when running across a very large internet or many gateways.
- +s** This option indicates that *xterm* should scroll synchronously.
- sb** This option indicates that some number of lines that are scrolled off the top of the window should be saved and that a scrollbar should be displayed so that those lines can be viewed. This option may be turned on and off from the "Modes" menu.
- +sb** This option indicates that a scrollbar should not be displayed.
- sf** This option indicates that Sun Function Key escape codes should be generated for function keys.
- +sf** This option indicates that the standard escape codes should be generated for function keys.
- si** This option indicates that output to a window should not automatically reposition the screen to the bottom of the scrolling region. This option can be turned on and off from the "Modes" menu.
- +si** This option indicates that output to a window should cause it to scroll to the bottom.
- sk** This option indicates that pressing a key while using the scrollbar to review previous lines of text should cause the window to be repositioned automatically in the normal position at the bottom of the scroll region.

- +sk** This option indicates that pressing a key while using the scrollbar should not cause the window to be repositioned.
- sl *number*** This option specifies the number of lines to save that have been scrolled off the top of the screen. The default is 64.
- t** This option indicates that *xterm* should start in Tektronix mode, rather than in VT102 mode. Switching between the two windows is done using the "Modes" menus.
- +t** This option indicates that *xterm* should start in VT102 mode.
- tm *string*** This option specifies a series of terminal setting keywords followed by the characters that should be bound to those functions, similar to the *stty* program. This is ignored when **-L** is given since *getty* resets the terminal. Allowable keywords include: *intr*, *quit*, *erase*, *kill*, *eof*, *eol*, *swtch*, *start*, *stop*, *brk*, *susp*, *dsusp*, *rpmt*, *flush*, *weras*, and *lnext*. Control characters may be specified as *^char* (e.g. *^c* or *^u*) and *^?* may be used to indicate delete.
- tn *name*** This option specifies the name of the terminal type to be set in the *TERM* environment variable. This terminal type must exist in the *termcap(5)* database and should have *li#* and *co#* entries.
- ut** This option indicates that *xterm* shouldn't write a record into the the system log file */etc/utmp*.
- +ut** This option indicates that *xterm* should write a record into the system log file */etc/utmp*.
- vb** This option indicates that a visual bell is preferred over an audible one. Instead of ringing the terminal bell whenever a Control-G is received, the window will be flashed.
- +vb** This option indicates that a visual bell should not be used.
- C** This option indicates that this window should receive console output. This is not supported on all systems.
- Scn** This option specifies the last two letters of the name of a pseudoterminal to use in slave mode. This allows *xterm* to be used as an input and output channel for an existing program and is sometimes used in specialized applications.

The following command line arguments are provided for compatibility with older versions. They may not be supported in the next release as the X Toolkit provides standard options that accomplish the same task.

- %geom** This option specifies the preferred size and position of the Tektronix window. It is shorthand for specifying the "tekGeometry" resource.

- #geom** This option specifies the preferred position of the icon window. It is shorthand for specifying the *"*iconGeometry"* resource.
- T string** This option specifies the title for *xterm*'s windows. It is equivalent to **-title**.
- n string** This option specifies the icon name for *xterm*'s windows. It is shorthand for specifying the *"*iconName"* resource. Note that this is not the same as the toolkit option **-name** (see below). The default icon name is the application name.
- r** This option indicates that reverse video should be simulated by swapping the foreground and background colors. It is equivalent to **-reversevideo** or **-rv**.
- w number** This option specifies the width in pixels of the border surrounding the window. It is equivalent to **-borderwidth** or **-bw**.
- L** This option indicates that *xterm* was started by *init*. In this mode, *xterm* does not try to allocate a new pseudoterminal as *init* has already done so. In addition, the system program *getty* is run instead of the user's shell. **This option has been superseded by the new *xdm* program; furthermore, this option should never be used by users when starting terminal windows.**

The following standard X Toolkit command line arguments are commonly used with *xterm*:

- bg color** This option specifies the color to use for the background of the window. The default is "white."
- bd color** This option specifies the color to use for the border of the window. The default is "black."
- bw number** This option specifies the width in pixels of the border surrounding the window.
- fg color** This option specifies the color to use for displaying text. The default is "black".
- fn font** This option specifies the font to be used for displaying normal text. The default is *fixed*.
- name name** This option specifies the application name under which resources are to be obtained, rather than the default executable file name. *Name* should not contain "." or "*" characters.

- title** *string*
This option specifies the window title string, which may be displayed by window managers if the user so chooses. The default title is the command line specified after the **-e** option, if any, otherwise the application name.
- rv** This option indicates that reverse video should be simulated by swapping the foreground and background colors.
- geometry** *geometry*
This option specifies the preferred size and position of the VT102 window; see X(1).
- display** *display*
This option specifies the X server to contact; see X(1).
- xrm** *resourcestring*
This option specifies a resource string to be used. This is especially useful for setting resources that do not have separate command line options.
- iconic** This option indicates that *xterm* should ask the window manager to start it as an icon rather than as the normal window.

X DEFAULTS

The program understands all of the core X Toolkit resource names and classes as well as:

iconGeometry (class **IconGeometry**)

Specifies the preferred size and position of the application when iconified. It is not necessarily obeyed by all window managers.

termName (class **TermName**)

Specifies the terminal type name to be set in the TERM environment variable.

title (class **Title**)

Specifies a string that may be used by the window manager when displaying this application.

ttyModes (class **TtyModes**)

Specifies a string containing terminal setting keywords and the characters to which they may be bound. This option is ignored when **-L** is given since *getty* resets the terminal. Allowable keywords include: *intr*, *quit*, *erase*, *kill*, *eof*, *eol*, *switch*, *start*, *stop*, *brk*, *susp*, *dsusp*, *rpmt*, *flush*, *weras*, and *lnext*. Control characters may be specified as *^char* (e.g. *^c* or *^u*) and *^?* may be used to indicate delete. This is very useful for overriding the default terminal settings without having to do an *stty* every time an *xterm* is started.

utmpInhibit (class **UtmpInhibit**)

Specifies whether or not *xterm* should try to record the user's terminal in */etc/utmp*.

sunFunctionKeys (class **SunFunctionKeys**)

Specifies whether or not Sun Function Key escape codes should be generated for function keys instead of standard escape sequences.

The following resources are specified as part of the "vt100" widget (class "VT100"):

allowSendEvents (class **AllowSendEvents**)

Specifies whether or not synthetic key and button events (generated using the X protocol SendEvent request) should be interpreted or discarded. The default is "false" meaning they are discarded. Note that allowing such events creates a very large security hole.

alwaysHighlight (class **AlwaysHighlight**)

Specifies whether or not *xterm* should always display a highlighted text cursor. By default, a hollow text cursor is displayed whenever the pointer moves out of the window or the window loses the input focus.

font (class **Font**)

Specifies the name of the normal font. The default is "vtsingle."

boldFont (class **Font**)

Specifies the name of the bold font. The default is "vtbold."

c132 (class **C132**)

Specifies whether or not the VT102 DECCOLM escape sequence should be honored. The default is "false."

charClass (class **CharClass**)

Specifies comma-separated lists of character class bindings of the form *[low-]high:value*. These are used in determining which sets of characters should be treated the same when doing cut and paste. See the section on specifying character classes.

curses (class **Curses**)

Specifies whether or not the last column bug in *curses*(3x) should be worked around. The default is "false."

background (class **Background**)

Specifies the color to use for the background of the window. The default is "white."

foreground (class **Foreground**)

Specifies the color to use for displaying text in the window. Setting the class name instead of the instance name is an easy way to have everything that would normally appear in the "text" color change color. The default is "black."

cursorColor (class **Foreground**)

Specifies the color to use for the text cursor. The default is "black."

geometry (class Geometry)

Specifies the preferred size and position of the VT102 window.

internalBorder (class BorderWidth)

Specifies the number of pixels between the characters and the window border. The default is 2.

jumpScroll (class JumpScroll)

Specifies whether or not jump scroll should be used. The default is "false".

logFile (class Logfile)

Specifies the name of the file to which a terminal session is logged. The default is "XtermLog.XXXXX" (where XXXXX is the process id of *xterm*).

logging (class Logging)

Specifies whether or not a terminal session should be logged. The default is "false."

logInhibit (class LogInhibit)

Specifies whether or not terminal session logging should be inhibited. The default is "false."

loginShell (class LoginShell)

Specifies whether or not the shell to be run in the window should be started as a login shell. The default is "false."

marginBell (class MarginBell)

Specifies whether or not the bell should be run when the user types near the right margin. The default is "false."

multiScroll (class MultiScroll)

Specifies whether or not asynchronous scrolling is allowed. The default is "false."

nMarginBell (class Column)

Specifies the number of characters from the right margin at which the margin bell should be run, when enabled.

pointerColor (class Foreground)

Specifies the color of the pointer. The default is "black."

pointerShape (class Cursor)

Specifies the name of the shape of the pointer. The default is "xterm."

reverseVideo (class ReverseVideo)

Specifies whether or not reverse video should be simulated. The default is "false."

reverseWrap (class ReverseWrap)

Specifies whether or not reverse-wraparound should be enabled. The default is "false."

saveLines (class SaveLines)

Specifies the number of lines to save beyond the top of the screen when a

scrollbar is turned on. The default is 64.

scrollBar (class **ScrollBar**)

Specifies whether or not the scrollbar should be displayed. The default is "false."

scrollInput (class **ScrollCond**)

Specifies whether or not output to the terminal should automatically cause the scrollbar to go to the bottom of the scrolling region. The default is "true."

scrollKey (class **ScrollCond**)

Specifies whether or not pressing a key should automatically cause the scrollbar to go to the bottom of the scrolling region. The default is "false."

signalInhibit (class **SignalInhibit**)

Specifies whether or not the entries in the "xterm X11" menu for sending signals to *xterm* should be disallowed. The default is "false."

tekGeometry (class **Geometry**)

Specifies the preferred size and position of the Tektronix window.

tekInhibit (class **TekInhibit**)

Specifies whether or not Tektronix mode should be disallowed. The default is "false."

tekStartup (class **TekStartup**)

Specifies whether or not *xterm* should start up in Tektronix mode. The default is "false."

titeInhibit (class **TiteInhibit**)

Specifies whether or not *xterm* should remove *ti* or *te* termcap entries (used to switch between alternate screens on startup of many screen-oriented programs) from the TERMCAP string.

translations (class **Translations**)

Specifies the key and button bindings for menus, selections, "programmed

visualBell (class **VisualBell**)

Specifies whether or not a visible bell (i.e. flashing) should be used instead of an audible bell when Control-G is received. The default is "false."

The following resources are specified as part of the "tek4014" widget (class "Tek4014")

width (class **Width**)

Specifies the width of the Tektronix window in pixels.

height (class **Height**)

Specifies the height of the Tektronix window in pixels.

The following resources are specified as part of the "menu" widget:

menuBorder (class MenuBorder)

Specifies the size in pixels of the border surrounding menus. The default is 2.

menuFont (class Font)

Specifies the name of the font to use for displaying menu items.

menuPad (class MenuPad)

Specifies the number of pixels between menu items and the menu border. The default is 3.

The following resources are useful when specified for the Athena Scrollbar widget:

thickness (class Thickness)

Specifies the width in pixels of the scrollbar.

background (class Background)

Specifies the color to use for the background of the scrollbar.

foreground (class Foreground)

Specifies the color to use for the foreground of the scrollbar. The "thumb" of the scrollbar is a simple checkerboard pattern alternating pixels for foreground and background color.

EMULATIONS

The VT102 emulation is fairly complete, but does not support the blinking character attribute nor the double-wide and double-size character sets. *Termcap*(5) entries that work with *xterm* include "xterm", "vt102", "vt100" and "ansi", and *xterm* automatically searches the termcap file in this order for these entries and then sets the "TERM" and the "TERMCAP" environment variables.

Many of the special *xterm* features (like logging) may be modified under program control through a set of escape sequences different from the standard VT102 escape sequences. (See the "*Xterm Control Sequences*" document in *xterm*(7))

The Tektronix 4014 emulation provides for four different font sizes and five different lines types. The Tektronix text and graphics commands are recorded internally by *xterm* and may be written to a file by sending the COPY escape sequence (or through the Tektronix menu; see below). The name of the file will be "COPYyy-MM-dd.hh:mm:ss", where yy, MM, dd, hh, mm and ss are the year, month, day, hour, minute and second when the COPY was performed (the file is created in the directory *xterm* is started in, or the home directory for a login *xterm*).

POINTER USAGE

Once the VT102 window is created, *xterm* allows you to select text and copy it within the same or other windows.

The selection functions are invoked when the pointer buttons are used with no modifiers, and when they are used with the "shift" key. The assignment of the functions described below to keys and buttons may be changed through the resource database; see **KEY/BUTTON BINDINGS** below.

Pointer button one (usually left) is used to save text into the cut buffer. Move the cursor to beginning of the text, and then hold the button down while moving the cursor to the end of the region and releasing the button. The selected text is highlighted and is saved in the global cut buffer and made the PRIMARY selection when the button is released. Double-clicking selects by words. Triple-clicking selects by lines. Quadruple-clicking goes back to characters, etc. Multiple-click is determined by the time from button up to button down, so you can change the selection unit in the middle of a selection. If the key/button bindings specify that an X selection is to be made, *xterm* will leave the selected text highlighted for as long as it is the selection owner.

Pointer button two (usually middle) 'types' (pastes) the text from the PRIMARY selection, if any, otherwise from the cut buffer, inserting it as keyboard input.

Pointer button three (usually right) extends the current selection. (Without loss of generality, that is you can swap "right" and "left" everywhere in the rest of this paragraph...) If pressed while closer to the right edge of the selection than the left, it extends/contracts the right edge of the selection. If you contract the selection past the left edge of the selection, *xterm* assumes you really meant the left edge, restores the original selection, then extends/contracts the left edge of the selection. Extension starts in the selection unit mode that the last selection or extension was performed in; you can multiple-click to cycle through them.

By cutting and pasting pieces of text without trailing new lines, you can take text from several places in different windows and form a command to the shell, for example, or take output from a program and insert it into your favorite editor. Since the cut buffer is globally shared among different applications, you should regard it as a 'file' whose contents you know. The terminal emulator and other text programs should be treating it as if it were a text file, i.e. the text is delimited by new lines.

The scroll region displays the position and amount of text currently showing in the window (highlighted) relative to the amount of text actually saved. As more text is saved (up to the maximum), the size of the highlighted area decreases.

Clicking button one with the pointer in the scroll region moves the adjacent line to the top of the display window.

Clicking button three moves the top line of the display window down to the pointer position.

Clicking button two moves the display to a position in the saved text that corresponds to the pointer's position in the scrollbar.

Unlike the VT102 window, the Tektronix window does not allow the copying of text. It does allow Tektronix GIN mode, and in this mode the cursor will change from an arrow to a cross. Pressing any key will send that key and the current coordinate of the cross cursor. Pressing button one, two, or three will return the letters 'l', 'm', and 'r', respectively. If the 'shift' key is pressed when a pointer button is pressed, the corresponding upper case letter is sent. To distinguish a pointer button from a key, the high bit of the character is set (but this bit is normally stripped unless the terminal mode is RAW; see `tty(4)` for details).

MENUS

Xterm has three menus, named **xterm**, **Modes**, and **Tektronix**. Each menu pops up under the correct combinations of key and button presses. Most menus are divided into two section, separated by a horizontal line. The top portion contains various modes that can be altered. A check mark appears next to a mode that is currently active. Selecting one of these modes toggles its state. The bottom portion of the menu are command entries; selecting one of these performs the indicated function.

The **xterm** menu pops up when the "control" key and pointer button one are pressed in a window. The Modes section contains items that apply to the VT102 and Tektronix windows. The **Secure Keyboard** mode should be used when typing in passwords or other sensitive data; see **SECURITY** below. Notable entries in the command section of the menu are the **Continue**, **Suspend**, **Interrupt**, **Hangup**, **Terminate** and **Kill** which sends the SIGCONT, SIGTSTP, SIGINT, SIGHUP, SIGTERM and SIGKILL signals, respectively, to the process group of the process running under *xterm* (usually the shell). The **Continue** function is especially useful if the user has accidentally typed CTRL-Z, suspending the process.

The **Modes** menu sets various modes in the VT102 emulation, and is popped up when the "control" key and pointer button two are pressed in the VT102 window. In the command section of this menu, the soft reset entry will reset scroll regions. This can be convenient when some program has left the scroll regions set incorrectly (often a problem when using VMS or TOPS-20). The full reset entry will clear the screen, reset tabs to every eight columns, and reset the terminal modes (such as wrap and smooth scroll) to their initial states just after *xterm* has finished processing the command line options. The Tektronix menu sets various modes in the Tektronix emulation, and is popped up when the "control" key and pointer button two are pressed in the Tektronix window. The current font size is checked in the modes section of the menu. The PAGE entry in the command section clears the Tektronix window.

SECURITY

X environments differ in their security consciousness. The servers provided by MIT use a host-based mechanism to control access to the server (see *xhost(1)*). If you enable access for a host, and other users are also permitted to run clients on that host, there is the possibility that someone will run an application that will attempt to use the basic services of the X protocol to snoop on your activities, and potentially capture a transcript of everything you type at the keyboard. This is of particular concern when you want to type in a password or other sensitive data. The best solution to this problem is for the industry to choose a standard authorization mechanism, with the necessary operating system support, and to incorporate this into the X protocol (which is already designed to handle such a mechanism). In the mean time, since passwords are most commonly typed to something running in an *xterm* window, a simple mechanism exists for protecting keyboard input in *xterm*.

The **xterm** menu (see **MENUS** above) contains a **Secure Keyboard** entry which, when enabled, ensures that all keyboard input is directed *only* to *xterm* (using the GrabKeyboard protocol request). When an application prompts you for a password

(or other sensitive data), you can enable **Secure Keyboard** using the menu, type in the data, and then disable **Secure Keyboard** using the menu again. Only one X client at a time can secure the keyboard, so when you attempt to enable **Secure Keyboard** it may fail. In this case, the bell will sound. If the **Secure Keyboard** succeeds, the foreground and background colors will be exchanged (as if you selected the **Reverse Video** entry in the **Modes** menu); they will be exchanged again when you exit secure mode. If the colors do *not* switch, then you should be *very* suspicious that you are being spoofed. If the application you are running displays a prompt before asking for the password, it is safest to enter secure mode *before* the prompt gets displayed, and to make sure that the prompt gets displayed correctly (in the new colors), to minimize the probability of spoofing. You can also bring up the menu again and make sure that a check mark appears next to the entry.

Secure Keyboard mode will be disabled automatically if your xterm window becomes iconified (or otherwise unmapped), or if you start up a reparenting window manager (that places a title bar or other decoration around the window) while in **Secure Keyboard** mode. (This is a feature of the X protocol not easily overcome.) When this happens, the foreground and background colors will be switched back and the bell will sound in warning.

CHARACTER CLASSES

Clicking the middle mouse button twice in rapid succession will cause all characters of the same class (e.g. letters, white space, punctuation) to be selected. Since different people have different preferences for what should be selected (for example, should filenames be selected as a whole or only the separate subnames), the default mapping can be overridden through the use of the *charClass* (class *CharClass*) resource.

This resource is simply a list of *range:value* pairs where the range is either a single number or *low-high* in the range of 0 to 127, corresponding to the ASCII code for the character or characters to be set. The *value* is arbitrary, although the default table uses the character number of the first character occurring in the set.

The default table is:

```
static int charClass[128] = {
/* NUL SOH STX ETX EOT ENQ ACK BEL */
  32, 1, 1, 1, 1, 1, 1, 1,
/* BS HT NL VT NP CR SO SI */
  1, 32, 1, 1, 1, 1, 1, 1,
/* DLE DC1 DC2 DC3 DC4 NAK SYN ETB */
  1, 1, 1, 1, 1, 1, 1, 1,
/* CAN EM SUB ESC FS GS RS US */
  1, 1, 1, 1, 1, 1, 1, 1,
/* SP ! " # $ % & ' */
  32, 33, 34, 35, 36, 37, 38, 39,
/* ( ) * + , - . */
  40, 41, 42, 43, 44, 45, 46, 47,
/* 0 1 2 3 4 5 6 7 */
}
```

```

48, 48, 48, 48, 48, 48, 48, 48,
/* 8 9 : ; < = > ? */
48, 48, 58, 59, 60, 61, 62, 63,
/* @ A B C D E F G */
64, 48, 48, 48, 48, 48, 48, 48,
/* H I J K L M N O */
48, 48, 48, 48, 48, 48, 48, 48,
/* P Q R S T U V W */
48, 48, 48, 48, 48, 48, 48, 48,
/* X Y Z [ \ ] ^ _ */
48, 48, 48, 91, 92, 93, 94, 48,
/* ' a b c d e f g */
96, 48, 48, 48, 48, 48, 48, 48,
/* h i j k l m n o */
48, 48, 48, 48, 48, 48, 48, 48,
/* p q r s t u v w */
48, 48, 48, 48, 48, 48, 48, 48,
/* x y z { | } ~ DEL */
48, 48, 48, 123, 124, 125, 126, 1);

```

For example, the string “33:48,37:48,45-47:48,64:48” indicates that the exclamation mark, percent sign, dash, period, slash, and ampersand characters should be treated the same way as characters and numbers. This is very useful for cutting and pasting electronic mailing addresses and Unix filenames.

KEY TRANSLATIONS

It is possible to rebind keys (or sequences of keys) to arbitrary strings for input, by changing the translations for the vt100 widget. Changing the translations for events other than key and button events is not expected, and will cause unpredictable behavior.

The actions available for key translations are:

- secure()** Toggles the **Secure Keyboard** mode; see **SECURITY**.
- insert()** Processes the key in the normal way; i.e. inserts the ASCII character code corresponding to the keysym found in the keyboard mapping table into the input stream.
- string(string)** Rebinds the key or key sequence to the string value; that is, inserts the string argument into the input stream. Quotation is necessary if the string contains whitespace or non-alphanumeric characters. If the string argument begins with the characters “0x”, it is interpreted as a hex character constant, and the corresponding character is sent in the normal way.
- keymap(name)** The **keymap** action takes a single string argument naming a resource to be used to dynamically define a new translation table; the name of the resource is obtained by appending the string “Keymap” to *name*. The keymap name **None** restores the

original translation table (the very first one; a stack is not maintained). Upper/lower case is significant.

insert-selection(*name*[,*name*]...)

Retrieves the value of the first (left-most) named selection that exists or cut buffer that is non-empty and inserts the value into the input stream. *Name* is the name of any selection, for example, **PRIMARY** or **SECONDARY**, or the name of a cut buffer: **CUT_BUFFER0**, ..., **CUT_BUFFER7**. Upper/lower case is significant.

For example, a debugging session might benefit from the following bindings:

```
*VT100.Translations: #\
  override <Key>F13: keymap(dbx)
*VT100.dbxKeymap.translations:\
  <Key>F14: keymap(None) \n\
  <Key>F17: string("next") string(0x0d) \n\
  <Key>F18: string("step") string(0x0d) \n\
  <Key>F19: string("continue") string(0x0d) \n\
  <Key>F20: string("print ") insert-selection(PRIMARY, CUT_BUFFER0)
```

KEY/BUTTON BINDINGS

Within the VT100 widget the key and button bindings for selecting text, pasting text, and activating the menus are controlled by the translation bindings. In addition to the actions listed above under **KEY TRANSLATIONS**, the following actions are available:

- mode-menu()** Posts one of the two mode menus, depending on which button is pressed.
- select-start()** Unselects any previously selected text and begins selecting new text.
- select-extend()** Continues selecting text from the previous starting position.
- start-extend()** Begins extending the selection from the farthest (left or right) edge.
- select-end**(*name*[,*name*]...) Ends the text selection. *Name* is the name of a selection, or the name of a cut buffer into which the text is to be copied. *Xterm* will assert ownership of all the selections named and will copy the text into each of the cut buffers. Upper/lower case is significant.
- ignore()** Quietly discards the key or button event.
- bell**([*volume*]) Rings the bell at the specified volume increment above/below the base volume.

The default bindings are:

		<KeyPress>:	insert() \n\
Ctrl	~Meta	<Btn1Down>:	mode-menu() \n\
	~Meta	<Btn1Down>:	select-start() \n\
	~Meta	<Btn1Motion>:	select-extend() \n\
Ctrl	~Meta	<Btn2Down>:	mode-menu() \n\
~Ctrl	~Meta	<Btn2Down>:	ignore() \n\
	~Meta	<Btn2Up>:	insert-selection(PRIMARY, CUT_BUFFER0) \n\
~Ctrl	~Meta	<Btn3Down>:	start-extend() \n\
	~Meta	<Btn3Motion>:	select-extend() \n\
	~Meta	<BtnUp>:	select-end(PRIMARY, CUT_BUFFER0) \n\
		<BtnDown>:	bell(0)

STARTING XTERM FROM INIT

Warning, this feature is now obsolete and may not be supported in future releases. Sites using this method should switch to *xdm* instead.

On operating systems such as 4.3bsd and Ultrix, the server and initial login window are normally started automatically by *init(8)*.

By convention, the pseudoterminal with the highest minor device number (e.g. */dev/ttyqf* and */dev/ptyqf*) is renamed for the lowest display number (e.g. */dev/ttyv0* and */dev/ptyv0*). Machines that have more than one display can repeat this process using *ttyqe* for *tyv1*, and so on.

Once the pseudoterminals are in place, a line similar to the following may be added to */etc/ttys* (replacing *Xqvss* with the appropriate server and putting it all on one line):

```
tyv0 "/usr/bin/X11/xterm -L -geom 80x24+1+1 -display :0"
      xterm on secure window="/usr/bin/X11/Xqvss :0"
```

Sites that used to run X10 should note that the colon in the server display number is required.

Although the release will install both the X server and *xterm* in */usr/bin/X11* by default, many sites choose to make a copy of both of these programs on the root partition (usually in */etc*) so that they may still be used even if the partition containing */usr/bin/X11* isn't mounted.

Some versions of *init* have relatively small program name buffer sizes and treat all sharp signs as comment delimiters. Sites that wish to list large numbers of options on the *xterm* line will need to write a small shell script to exec the long *xterm* line. The best solution, of course, is to use *xdm*.

OTHER FEATURES

Xterm automatically highlights the window border and text cursor when the pointer enters the window (selected) and unhighlights them when the pointer leaves the window (unselected). If the window is the focus window, then the window is highlighted no matter where the pointer is.

In VT102 mode, there are escape sequences to activate and deactivate an alternate screen buffer, which is the same size as the display area of the window. When activated, the current screen is saved and replaced with the alternate screen. Saving of lines scrolled off the top of the window is disabled until the normal screen is restored. The *termcap*(5) entry for *xterm* allows the visual editor *vi*(1) to switch to the alternate screen for editing, and restore the screen on exit.

In VT102 or Tektronix mode, there are escape sequences to change the name of the windows and to specify a new log file name.

ENVIRONMENT

Xterm sets the environment variables "TERM" and "TERMCAP" properly for the size window you have created. It also uses and sets the environment variable "DISPLAY" to specify which bit map display terminal to use. The environment variable "WINDOWID" is set to the X window id number of the *xterm* window.

SEE ALSO

resize(1), *X*(1), *pty*(4), *tty*(4), *xterm*(7).

BUGS

The **-L** option is no longer needed as the new *xdm* display manager system handles logging in in a much cleaner way. No more messing around with trying to match colors in */etc/tty*s or worrying about an unwanted login window. **This option may be removed in future releases.**

Xterm will hang forever if you try to paste too much text at one time. It is both producer and consumer for the *pty* and can deadlock.

Variable-width fonts are not handled reasonably.

This program still needs to be rewritten. It should be split into very modular sections, with the various emulators being completely separate widgets that don't know about each other. Ideally, you'd like to be able to pick and choose emulator widgets and stick them into a single control widget.

The focus is considered lost if some other client (e.g., the window manager) grabs the pointer; it is difficult to do better without an addition to the protocol.

There needs to be a dialog box to allow entry of log file name and the COPY file name.

Many of the options are not resettable after *xterm* starts.

The Tek widget does not support key/button re-binding.

This manual page is too long. There should be a separate users manual defining all of the non-standard escape sequences.

All programs should be written to use X directly; then we could eliminate this program.

COPYRIGHT

Copyright 1988, 1989, Massachusetts Institute of Technology.
See *X(1)* for a full statement of rights and permissions. Notes about Security are
Copyright 1989, Apollo Computer Inc.

AUTHIORS

Far too many people, including:

Loretta Guarino Reid (DEC-UEG-WSL), Joel McCormack (DEC-UEG-WSL), Terry
Weissman (DEC-UEG-WSL), Edward Moy (Berkeley), Ralph R. Swick (MIT-
Athena), Mark Vandevoorde (MIT-Athena), Bob McNamara (DEC-MAD), Jim Get-
tys (MIT-Athena), Bob Scheifler (MIT X Consortium), Doug Mink (SAO), Steve
Pitschke (Stellar), Ron Newman (MIT-Athena), Jim Fulton (MIT X Consortium),
Dave Serisky (HP), Brian Holt (Apollo)

NAME

yacc – yet another compiler-compiler

SYNOPSIS

yacc [**-vd**] *grammar*

DESCRIPTION

yacc converts a context-free grammar into a set of tables for a simple automaton which executes an LR parsing algorithm. The grammar may be ambiguous; **yacc** uses specified precedence rules to eliminate such problems.

The output file, **y.tab.c**, must be compiled by the C compiler to produce a program **yyparse**. This program must be loaded with the lexical analyzer program, **yylex**, as well as **main** and **yyerror**, an error-handling routine. You must supply these routines. **lex(1)** helps create lexical analyzers usable by **yacc**.

OPTIONS

- v** Prepare the **y.output** file, which contains a description of the parsing tables and a report on conflicts generated by ambiguities in the grammar.
- d** Generate the **y.tab.h** file with the define statements that associate the token codes supplied by **yacc** with the user-declared token names. This allows source files other than **y.tab.c** to access the token codes.

FILES

y.output	
y.tab.c	
y.tab.h	defines for token names
yacc.tmp, yacc.acts	temporary files
/usr/lib/yaccpar	parser prototype for C programs

DIAGNOSTICS

yacc reports, on the standard output, the number of reduce-reduce and shift-reduce conflicts. It also lets you know if some rules cannot be reached from the start symbol. The **y.output** file contains a more detailed report.

BUGS

Because filenames are fixed, only one **yacc** process can be active in a given directory at a time.

SEE ALSO

lex(1)

NAME

yes – be repetitively affirmative

SYNOPSIS

yes [*expletive*]

DESCRIPTION

yes repeatedly outputs “y”, or if *expletive* is given, that is output repeatedly. Terminate **yes** with an interrupt.

THE

PROGRESS

OF

THE

ARTS

AND

MANUFACTURES

IN

THE

UNITED

STATES

OF

AMERICA

FROM

1790

TO

1860

BY

JOHN

W. FOSTER

OF

THE

AMERICAN

ANTI-SLAVERY

SOCIETY

NEW

YORK

1860

Section 6

Games

NAME

intro – introduction to games

DESCRIPTION

This section describes the recreational and educational programs found in the directory */usr/games*.

NAME

domain – Domain/OS-specific games

DESCRIPTION

While providing all of the significant functionality of 4.3BSD, Domain/OS BSD actually represents only a subset of the greater functionality of Domain/OS. Furthermore, Domain/OS BSD omits some features of 4.3BSD, that are irrelevant to Apollo® workstations. The following paragraphs list additional games available in the Domain/OS `/usr/games` directory.

Domain/OS Additions to the BSD Environment

The `/usr/games` directory includes standard 4.3BSD games, System V games, and Domain/OS-specific games. Pages that describe Domain/OS-specific games have the heading, "Domain/OS BSD"; pages documenting standard UNIX games are identified with the heading "BSD".

bgcolor	Make interesting background colors
bj	The game of blackjack
btlf fortune	Bell Telephone Labs' version of fortune
btlgammon	Bell Telephone Labs' version of fortune
btlangman	Bell Telephone Labs' version of hangman
craps	The game of craps
dmoire	Domain/Dialogue-based moire generator
factor	Factoring program
flake	Induce terminal dandruff
mastermind	Mastermind guessing game
maze	Generate a maze
melt	"Melt" the screen
moo	Guessing game
primes	Print prime numbers
puzzle	A puzzle game
random	Random number generator
revscr	Reverse screen
scramble	Turn your screen into a scramble puzzle
teachgammon	Teach the game of backgammon
ttt	The game of tic-tac-toe
vine	Grow vines

Unsupported Games

The following games from 4.3BSD are not supported:

aardvark
adventure
chess
ching
doctor
rogue
zork

SEE ALSO

domain(1), intro(6), domain(7), domain(8) domain(.8) "

NAME

arithmetic – provide drill in number facts

SYNOPSIS

`/usr/games/arithmetic [ +-x/ ] [ range ]`

DESCRIPTION

arithmetic presents simple arithmetic problems, and waits for you to type an answer. If the answer is correct, it replies "Right!", and supplies a new problem. If the answer is wrong, it replies "What?", until you respond correctly.

The first optional argument determines the kind of problem to be generated. A plus sign (+), minus sign (-), lowercase x, and a slash (/) produce addition, subtraction, multiplication, and division problems respectively. Specifying more than one of these characters on a command line generates a variety of problem types, all mixed in random order. Specifying any characters other than the four mentioned here also produces a random mix of problem types. If you specify no argument to **arithmetic**, subtraction problems appear by default.

The second optional argument is *range*, a decimal number. If used, all addends, subtrahends, differences, multiplicands, divisors, and quotients will be less than or equal to this number. The default *range* is 10.

At the start, all numbers less than or equal to *range* are equally likely to appear. If the respondent makes a mistake, the numbers in the problem which was missed become more likely to reappear.

Every twenty problems, it publishes statistics on correctness and the time required to answer. Specifically, the program tells you the number of correct and incorrect answers that you have given, as well as the total percentage of those correct. It also tells you how much time (in seconds) has elapsed, and the average number of seconds it took you to answer each problem. For example, the program may output something like this:

```
Rights 20; Wrongs 1; Score 95%
Total time 50 seconds; 2.5 seconds per problem
```

To quit the program, type an interrupt (usually CTRL/C).

NOTES

As a matter of educational philosophy, **arithmetic** does not supply correct answers, since the learner should be able to calculate them. Thus, it does not try to teach number facts, but instead serves as a drill program for those just past the first learning stage of arithmetic. Usually, the most relevant statistic it provides is time per problem, not percent correct.

NAME

backgammon – the game of backgammon

SYNOPSIS

/usr/games/backgammon [*options*] [*file*]

DESCRIPTION

This program lets you play backgammon against the computer or against a friend. All commands consist of only one letter, so you don't need to type a carriage return except at the end of a move. The program is mostly self documenting; typing a question mark (?) will usually get some help. If you answer **y** when the program asks if you want the rules, you will get text explaining the rules of the game, some hints on strategy, instructions on how to use the program, and a tutorial consisting of a practice game against the computer.

OPTIONS

- n** Don't ask for rules or instructions.
- r** Player is red (implies **n**).
- w** Player is white (implies **n**).
- b** Two players, red and white (implies **n**). **TP -pr** Print the board before red's turn.
- pw** Print the board before white's turn.
- pb** Print the board before both players' turns.

Several arguments may be concatenated together. If your terminal has capabilities for direct cursor movement, **backgammon** "fixes" the board after each move so that you need not reprint the board each time. (In this case, all **-pr**, **-pw**, and **-pb** options are ignored.)

COMMANDS

When the program prompts by typing only your color, type a space or carriage return to roll, or

- d** to double
- p** to print the board
- q** to quit
- s** to save the game for later

When the program prompts with "Move:", type

- p** to print the board
 - q** to quit
 - s** to save the game
- or a move, which is a sequence of
- s-f** move from **s** to **f**

s/r move one man on **s** the roll **r**

separated by commas or spaces and ending with a newline. Available abbreviations are

s-f1-f2 means **s-f1,f1-f2**

s/r1r2 means **s/r1,s/r2**

Use **b** for bar and **h** for home, or 0 or 25 as appropriate.

FILES

/etc/termcap	terminal capability database
/usr/games/teachgammon	rules and tutorial

NAME

banner – print large banner on printer

SYNOPSIS

`/usr/games/banner [ -wn ] [ message ]`

DESCRIPTION

banner prints a large, high quality banner on the standard output. If *message* is omitted, **banner** prompts for and reads one line of its standard input. The output should be printed on a hardcopy device up to 132 columns wide, with no breaks between the pages.

OPTIONS

-wn Size output for device of width *n*. If *n* is omitted, a value of 80 is assumed.

BUGS

Several ASCII characters are not defined, notably <, >, [,], \, ^, _, {, }, |, and ~. Also, the characters ", ', and & are funny looking (but in a useful way.)

The **-w** option is implemented by skipping some rows and columns. The smaller it gets, the grainier the output. Sometimes it runs letters together.

NAME

battlestar – a tropical adventure game

SYNOPSIS

battlestar [-r]

DESCRIPTION

battlestar is an adventure game in the classic style. However, it's slightly less of a puzzle and more a game of exploration. There are a few magical words in the game, but on the whole, simple English should suffice to make one's desires understandable to the parser.

THE SETTING

In the days before the darkness came, when battlestars ruled the heavens...

Three He made and gave them to His daughters,
Beautiful nymphs, the goddesses of the waters.
One to bring good luck and simple feats of wonder,
Two to wash the lands and churn the waves asunder,
Three to rule the world and purge the skies with thunder.

In those times great wizards were known and their powers were beyond belief. They could take any object from thin air, and, uttering the word 'su' could disappear.

In those times men were known for their lust of gold and desire to wear fine weapons. Swords and coats of mail were fashioned that could withstand a laser blast.

But when the darkness fell, the rightful reigns were toppled. Swords and helms and heads of state went rolling across the grass. The entire fleet of battlestars was reduced to a single ship.

COMMANDS

For commands which manipulate objects, the "shadow" of the next word stays around if you want to take advantage of it: that is, saying **take knife** and then **drop** will drop the knife you just took.

take Take an object

drop Drop an object

wear Wear an object you are holding

draw Carry an object you are wearing

puton Take an object and wear it

take off Draw an object and drop it

throw *object direction*

Throw an object in the specified direction

N S E W Move in one of the four compass directions. You may use these commands only if you have a compass.

- R L A B** Move right/left/ahead/back. Directions printed in room descriptions are always printed using these relative directions.
- inven** Display inventory
- save** Save the game in a file named **Bstar**. Saved games can be restarted using the **-r** option.
- !** Escape to a shell
- score** Display current score

BUGS
Countless.

NAME

bcd – convert to antique media

SYNOPSIS

/usr/games/bcd text

DESCRIPTION

bcd converts the literal *text* into a form familiar to old-timers.

SEE ALSO

dd(1)

NAME

bgcolor – make interesting background colors

SYNOPSIS

/usr/games/bgcolor [*options*]

DESCRIPTION

bgcolor changes the display's background colors.

OPTIONS

- s[lot] *num*** Specify the color as value from the color table. The default value is 4.
- c[olor] *rgbval*** Specify the color as an RGB value.
- f[ade]** Change color continuously, cycling through RGB values.
- i[ncrement] *num*** Change RGB value by *num* for each color update induced by the **-fade** option. Higher values cause the background color to cycle more quickly. Default value is 1.
- d[evice] *name*** Set the device to be used by **bgcolor**. Possible values for *name* are **borrow** (use entire display), **borrow_nc** (same as **borrow** but doesn't clear screen first), **direct** (in current window only), and **bg** (use display background).
- nb** Display no borders around target window.

BUGS

bgcolor produces unpredictable results on monochrome displays.

NAME

bj – the game of blackjack

SYNOPSIS

/usr/games/bj

DESCRIPTION

bj is a serious attempt at simulating the dealer in the game of blackjack (or twenty-one) as might be found in Reno. The following rules apply:

The bet is \$2 every hand.

A player "natural" (black jack) pays \$3. A dealer natural loses \$2. Simultaneous dealer and player naturals is a "push" (no money exchange).

If the dealer has an ace up, you can make an "insurance" bet against the chance of a dealer natural. If this bet is not taken, play resumes as normal. If the bet is taken, it is a side bet where you win \$2 if the dealer has a natural, and lose \$1 if the dealer does not.

If dealt two cards of the same value, you can "double", that is, play two hands, each with one of these cards. The bet also doubles (\$2 on each hand).

If a dealt hand totals 10 or 11, you may "double down". This means that you may double the bet (\$2 to \$4) and receive exactly one more card on that hand.

Under normal play, you may "hit" (draw a card) as long as your total isn't over twenty-one. If you "bust" (go over twenty-one), the dealer wins the bet.

When you "stand" (decide not to hit), the dealer hits until attaining a total of seventeen or more. If the dealer busts, you win the bet.

If both you and the dealer stand, the one with the largest total wins. A tie is a push.

The machine deals and keeps score. The following questions are asked at appropriate times. You must answer each question by a **y** and a carriage return for "yes", or just a carriage return for "no".

? (This means "do you want a hit?")

Insurance?

Double down?

Every time the deck is shuffled, the dealer so states and the "action" (total bet) and "standing" (total won or lost) is printed. To exit, type an interrupt and the action and standing are printed.

NAME

boggle – play the game of boggle

SYNOPSIS

/usr/games/boggle [+([+])]

DESCRIPTION

This program is intended for people wishing to sharpen their skills at Boggle (TM Parker Bros.). If you invoke the program with 4 arguments of 4 letters each (*e.g.*, “**boggle appl epie moth erhd**”), the program forms the obvious Boggle grid and lists all the words from **/usr/dict/words** found therein. If you invoke the program without arguments, it will generate a board for you, let you enter words for 3 minutes, and then tell you how well you did relative to **/usr/dict/words**.

The object of Boggle is to find, within 3 minutes, as many words as possible in a 4 by 4 grid of letters. Words may be formed from any sequence of 3 or more adjacent letters in the grid. The letters may join horizontally, vertically, or diagonally. However, no position in the grid may be used more than once within any one word. In competitive play amongst humans, each player is given credit for those of his words which no other player has found.

In interactive play, enter your words separated by spaces, tabs, or newlines. A bell will ring when there is 2:00, 1:00, 0:10, 0:02, 0:01, and 0:00 time left. You may complete any word started before the expiration of time. You can surrender before time is up by hitting an interrupt key. While entering words, your erase character is only effective within the current word and your line kill character is ignored.

Advanced players may wish to invoke the program with one or two plus signs (+) as the argument. The first + removes the restriction that positions can only be used once in each word. The second + causes a position to be considered adjacent to itself as well as its (up to) 8 neighbors.

FILES

/usr/dict/words

NAME

btlfortune – print a random comment

SYNOPSIS

/usr/games/btlfortune

DESCRIPTION

This is Bell Telephone Labs' version of **fortune**. **btlfortune** prints a fortune, anecdote, saying, or other random comment. All lines are derived from the default fortune database in **/usr/games/lib/btlfortunes**.

NAME

btlgammon – the game of backgammon

SYNOPSIS

/usr/games/btlgammon

DESCRIPTION

This is Bell Telephone Labs' version of **backgammon**. It will ask whether you need instructions.

NAME

btlhangman – guess the word

SYNOPSIS

/usr/games/btlhangman [*arg*]

DESCRIPTION

This is Bell Telephone Labs' version of **hangman**. **btlhangman** chooses a word at least seven letters long from a dictionary. You must guess letters, one at a time, until you guess the word.

The optional argument *arg* names an alternate dictionary.

FILES

/usr/games/lib/w2006 dictionary

NOTES

Hyphenated compounds are run together.

NAME

canfield, **cfscores** – the solitaire card game canfield

SYNOPSIS

```
/usr/games/canfield
/usr/games/cfscores [ -a ] [ user ]
```

DESCRIPTION

If you have never played solitaire before, it is recommended that you consult a solitaire instruction book. In Canfield, tableau cards may be built on each other downward in alternate colors. An entire pile must be moved as a unit in building. Top cards of the piles are available to be played on foundations, but never into empty spaces.

Spaces must be filled from the stock. The top card of the stock also is available to be played on foundations or built on tableau piles. After the stock is exhausted, tableau spaces may be filled from the talon and the player may keep them open until he wishes to use them.

Cards are dealt from the hand to the talon by threes and this repeats until there are no more cards in the hand or the player quits. To have cards dealt onto the talon the player types **ht** for his move. Foundation base cards are also automatically moved to the foundation when they become available.

The command **c** causes **canfield** to maintain card counting statistics on the bottom of the screen. When properly used this can greatly increase one's chances of winning.

The rules for betting are somewhat less strict than those used in the official version of the game. The initial deal costs \$13. You may quit at this point or inspect the game. Inspection costs \$13 and allows you to make as many moves as possible without moving any cards from your hand to the talon. (The initial deal places three cards on the talon; if all these cards are used, three more are made available.) Finally, if the game seems interesting, you must pay the final installment of \$26. At this point you are credited at the rate of \$5 for each card on the foundation; as the game progresses you are credited with \$5 for each card that is moved to the foundation. Each run through the hand after the first costs \$5. The card counting feature costs \$1 for each unknown card that is identified. If the information is toggled on, you are only charged for cards that became visible since it was last turned on. Thus the maximum cost of information is \$34. Playing time is charged at a rate of \$1 per minute.

With no arguments, the program **cfscores** prints out the current status of your canfield account. If a user name is specified, it prints out the status of their canfield account. If the **-a** flag is specified, it prints out the canfield accounts for all users that have played the game since the database was set up.

FILES

<code>/usr/games/canfield</code>	the game itself
<code>/usr/games/cfscores</code>	the database printer
<code>/usr/games/lib/cfscores</code>	the database of scores

BUGS

It is impossible to cheat.

NAME

craps – the game of craps

SYNOPSIS

/usr/games/craps

DESCRIPTION

craps is a form of the game of craps that is played in Las Vegas. The program simulates the *roller*, while you place bets. At any time, you may choose to bet with the roller or with the *House*. A bet of a negative amount is taken as a bet with the House; any other bet is a bet with the roller.

At the start of the game, you have a "bankroll" of \$2,000. The program begins prompting with:

"bet?"

The bet can be all or part of your bankroll. Any bet over the total bankroll is rejected and the program continues prompting until a proper bet is made.

Once the bet is accepted, the roller throws the dice. The following rules apply (you win or lose, depending on whether the bet is placed with the roller or with the House; the odds are even). The first roll is the roll immediately following a bet:

1. On the first roll:

7 or 11 -- wins for the roller;

2, 3, or 12 -- wins for the House;

any other number is the point, so roll again (Rule 2 applies)

2. On subsequent rolls:

point -- roller wins;

7 -- House wins;

any other number -- roll again.

If you lose your entire bankroll, the House offers to lend you an additional \$2,000. The program prompts as follows:

"marker?"

A **yes** (or **y**) consummates the loan. Any other reply terminates the game.

If you owe the House money, the House reminds you, before you can place a bet, how many markers are outstanding.

If, at any time, you have outstanding markers and your bankroll exceeds \$2,000, the House asks:

"Repay marker?"

A reply of **yes** (or **y**) indicates your willingness to repay the loan. If only 1 marker is outstanding, the debt is immediately repaid. However, if more than 1 marker is outstanding, the House asks:

“How many?”

markers you want to repay. If you enter an invalid number or just a carriage return, the program prints an appropriate message and prompts with

“How many?”

until you provide a valid number.

If you accumulate 10 markers (a total of \$20,000 borrowed from the House), the program tells you and then exits.

Should your bankroll exceed \$50,000, the House automatically deducts from it the total amount of money needed to pay off all outstanding markers.

If you accumulate \$100,000 or more, you break the bank. The program then prompts:

“New game?”

to give the House a chance to win back its money.

The program usually considers any reply other than a **yes** to be a **no**. Exceptions to this are when the program asks you if you want to place a bet (i.e., bet?) and when it asks how many markers you want to pay off (i.e., How many?).

To exit, send an interrupt (CTRL/I). Before exiting, the program tells you whether you won, lost, or broke even.

MISCELLANEOUS

The random number generator for the die numbers uses the seconds from the time of day. Depending on system usage, these numbers, at times, may seem strange but occurrences of this type in a real dice situation are not uncommon.

NAME

cribbage – the card game cribbage

SYNOPSIS

`/usr/games/cribbage [ -req ] name ...`

DESCRIPTION

cribbage allows you to play the card game cribbage. The program plays one hand and you play the other. At the beginning of the game, the program asks if you need to see the rules of the game. If so, it will print out the appropriate section from *According to Hoyle* with `more(1)`.

cribbage first asks you whether you wish to play a short game ("once around", to 61) or a long game ("twice around", to 121). A response of 's' results in a short game; any other response results in a long game.

At the start of the first game, the program asks you to cut the deck to determine who gets the first crib. You should respond with a number between 0 and 51, indicating how many cards down the deck is to be cut. Whoever cuts the lower ranked card gets the first crib. If more than one game is played, the loser of the previous game gets the first crib in the current game.

For each hand, the program first prints your hand, whose crib it is, and then asks you to discard two cards into the crib. The cards are prompted for one per line, and are typed as explained below.

After discarding, the program cuts the deck (if it is your crib) or asks you to cut the deck (if it's the program's crib). In the latter case, the appropriate response is a number from 0 to 39 indicating how far down the remaining 40 cards are to be cut.

After cutting the deck, play starts with the non-dealer (the player who doesn't have the crib) leading the first card. Play continues until all cards are exhausted. The program keeps track of the scoring of all points and the total of the cards on the table.

After play, the hands are scored. The program asks you to score your hand (and the crib, if yours) by printing out the appropriate cards (and the cut card enclosed in brackets). Play continues until one player reaches the game limit (61 or 121).

A carriage return when a numeric input is expected is equivalent to typing the lowest legal value; when cutting the deck, this is equivalent to choosing the top card.

Cards are specified as rank followed by suit. You may specify ranks by typing a one-character identifier, or by spelling out the rank as a word. Following are valid entries:

- a ace
- 2 two
- 3 three
- 4 four
- 5 five
- 6 six
- 7 seven
- 8 eight

9 nine
t ten
j jack
q queen
k king

Suits may be specified as:

s spaces
h hearts
d diamonds
c clubs

A card may be specified as: <rank> " " <suit> or <rank> " of " <suit>. If the single letter rank and suit designations are used, the space separating the suit and rank may be left out. Also, if only one card of the desired rank is playable, typing the rank is sufficient. For example, if your hand is "2H, 4D, 5C, 6H, JC, KD" and you want to discard the king of diamonds, any of the following could be typed: "k", "king", "kd", "k d", "k of d", "king d", "king of d", "k diamonds", "k of diamonds", "king diamonds", or "king of diamonds".

OPTIONS

- e If you make a mistake scoring your hand or crib, provide an explanation of the correct score. (This is especially useful for beginning players.)
- q Print a shorter form of all messages. (This is only recommended for users who have already played the game without specifying this option.)
- r Instead of asking the player to cut the deck, randomly cut the deck.

NAME

dmoire – Domain/Dialogue-based moire generator

SYNOPSIS

/usr/games/dmoire [-w | -b | -i] [-inv -nb -nd] [-fg color] [-bg color]

DESCRIPTION

dmoire creates moire patterns by moving simple geometric shapes across the display and allowing these shapes to overlap. By default, **dmoire** draws in the background of the screen, but will also use a separate window, borrow the display, or even the window its own menus are in.

This program was adapted from a public domain desk accessory.

OPTIONS

-w[indow]	Draw moires in a new window.
-b[orrow]	Borrow the display and draw a simple moire.
-i[ndialog]	Draw inside the menu interface.
-inv[erse]	Invert the background and foreground colors.
-nb[order]	Use with -w to remove the window border.
-nd[ialog]	Do not supply dialog menus.
-fg	Specify a foreground color as an index into your color map (0-255).
-bg	Specify a background color.

BUGS

The **-i** option is a hack which doesn't work very well.

NAME

factor – factoring program

SYNOPSIS

/usr/games/factor [*number*]

DESCRIPTION

factor prints the prime factors of the integer *number* and then exits. If you run **factor** without an argument, it reads lines from the standard input, factoring each number given. Entering a blank line or a non-numeric character causes **factor** to exit.

NAME

fish – play “Go Fish”

SYNOPSIS

/usr/games/fish

DESCRIPTION

fish plays the game of “Go Fish,” a childrens’ card game. The object is to accumulate “books” of 4 cards with the same face value. The players alternate turns; each turn begins with one player selecting a card from his hand, and asking the other player for all cards of that face value. If the other player has one or more cards of that face value in his hand, he gives them to the first player, and the first player makes another request.

Eventually, the first player asks for a card which is not in the hand of the second player, who replies “GO FISH!” The first player then draws a card from the pool of undealt cards. If this is the card he had last requested, he draws again. When a book is made, either through drawing or requesting, the cards are laid down and no further action takes place with that face value.

To play the computer, simply make guesses by typing **a, 2, 3, 4, 5, 6, 7, 8, 9, 10, j, q,** or **k** when asked. Pressing <RETURN> gives you information about the size of your opponent’s hand and the pool, and tells you about your opponent’s books. Entering the command **p** in place of your first guess puts you into the “pro” level. The default is not very difficult.

NAME

flake – induce terminal dandruff

SYNOPSIS

/usr/games/flake

DESCRIPTION

flake causes bits of the screen display to fall off. Type CTRL/Q to put a stop to it.

NAME

fortune – print a random, hopefully interesting, adage

SYNOPSIS

/usr/games/fortune [-] [-wslao] [-m *pattern*] [*file*]

DESCRIPTION

fortune with no arguments prints out a random adage. You may specify an alternate file of adages from which to read. This file must be created by **strfile(6)**. Only one such file may be named in a single command line; subsequent ones are ignored.

OPTIONS

- w** Wait before termination for an amount of time calculated from the number of characters in the message. This is useful if **fortune** is executed as part of the logout procedure to guarantee that the message can be read before the screen is cleared.
- s** Short apothegms only.
- l** Long dicta only.
- o** Choose from an alternate list of aphorisms, often used for potentially offensive ones.
- a** Choose from either list of maxims.
- m *pattern*** Print all fortunes which match the regular expression *pattern*. See **regexp(3)**.

FILES

/usr/games/lib/fortunes.dat

SEE ALSO

regexp(3), **strfile(6)**

NAME

hangman – Computer version of the game hangman

SYNOPSIS

`/usr/games/hangman`

DESCRIPTION

In *hangman*, the computer picks a word from the on-line word list and you must try to guess it. The computer keeps track of which letters have been guessed and how many wrong guesses you have made on the screen in a graphic fashion.

FILES

`/usr/dict/words` On-line word list

NAME

hunt – a multi-player multi-terminal game

SYNOPSIS

`/usr/games/hunt [-q] [-m] [hostname] [-l name]`

DESCRIPTION

The object of the game **hunt** is to kill off the other players. There are no rooms, no treasures, and no monsters. Instead, you wander around a maze, find grenades, trip mines, and shoot down walls and players. The more players you kill before you die, the better your score is.

hunt normally looks for an active game on the local network; if none is found, it starts one up on the local host. You may specify the location of the game by providing a *hostname* argument.

hunt only works on screens with at least 24 lines, 80 columns, and cursor addressing. The screen is divided into 3 areas. On the right hand side is the status area. It shows you how much damage you've sustained, how many charges you have left, who's in the game, who's scanning (shown by an asterisk in front of the name), who's cloaked (shown by a plus sign in front of the name), and other players' scores. The rest of the screen is taken up by your map of the maze, except for the 24th line, which is used for longer messages that don't fit in the status area.

The screen symbols used in **hunt** are as follows:

```
-|+   - walls
/\     - diagonal (deflecting) walls
#      - doors (dispersion walls)
;      - small mine
g      - large mine
:      - shot
o      - grenade
O      - satchel charge
@      - bomb
s      - small slime bomb
$      - big slime bomb
><^v  - you facing right, left, up, or down
}{i!   - other players facing right, left, up, or down
*      - explosion
\|/
-* -   - grenade and large mine explosion
/|\
```

Satchel and bomb explosions are larger than grenades (5x5, 7x7, and 3x3 respectively).

Your score is the ratio of number of kills to number of times you entered the game and is only kept for the duration of a single session of **hunt**.

hunt normally drives up the load average to be about (number_of_players + 0.5) greater than it would be without a **hunt** game executing. A limit of three players per

host and nine players total is enforced by **hunt**.

COMMANDS

h j k l	Move left/down/up/right. (Note that hunt uses the same keys as vi(1) for movement.) To change the direction you're facing, use the upper case version of the key.
y u b n	Move in a diagonal direction.
f	Fire in the direction you're facing. Takes 1 charge.
g	Throw grenade in the direction you're facing. Takes 9 charges.
F	Throw satchel charge. Takes 25 charges.
G	Throw bomb. Takes 49 charges.
o	Throw small slime bomb. Takes 15 charges.
O	Throw big slime bomb. Takes 30 charges.
s	Scan (show where other players are). Takes 1 charge.
c	Cloak (hide from scanners). Takes 1 charge.
CTRL/L	Redraw screen
q	Quit

OPTIONS

-l name	Specify player name. This command syntax was chosen for rlogin/rsh compatibility.
-m	Enter game as monitor: you can see the action but may not participate.
-q	Query the network and report if an active game is found. This is useful for .login scripts.

HELPFUL HINTS

- You can only fire in the direction you are facing.
- You can only fire three shots in a row, then the gun must cool.
- A shot only affects the square it hits.
- Shots and grenades move 5 times faster than you do.
- To stab someone, you must face that player and move at them.
- Stabbing does 2 points worth of damage and shooting does 5 points.
- Slime does 5 points of damage each time it hits.
- You start with 15 charges and get 5 more for every new player.
- A grenade affects the nine squares centered about the square it hits.
- A satchel affects the twenty-five squares centered about the square it hits.
- A bomb affects the forty-nine squares centered about the square it hits.
- Slime affects all squares it oozes over (15 or 30 respectively).
- One small mine and one large mine is placed in the maze for every new player. A mine has a 5% probability of tripping when you walk directly at it; 50% when going sideways on to it; 95% when backing up on to it. Tripping a mine costs you 5 points or 10 points respectively. Defusing a mine is worth 1 charge or 9 charges respectively.

- You cannot see behind you.
- Scanning lasts for 20*(number of players) turns. Scanning takes 1 ammo charge, so don't waste all your charges scanning.
- Cloaking lasts for 20 turns.
- Whenever you kill someone, you get 2 more damage capacity points and 2 damage points taken away.
- Maximum type-ahead is 5 characters.
- A shot destroys normal (*i.e.*, non-diagonal, non-door) walls.
- Diagonal walls deflect shots and change orientation.
- Doors disperse shots in random directions (up, down, left, right).
- Diagonal walls and doors cannot be destroyed by direct shots but may be destroyed by an adjacent grenade explosion.
- Slime goes around walls, not through them.
- Walls regenerate, reappearing in the order they were destroyed. One percent of the regenerated walls will be diagonal walls or doors. When a wall is generated directly beneath a player, he is thrown in a random direction for a random period of time. When he lands, he sustains damage (up to 20 percent of the amount of damage he had before impact); that is, the less damage he had, the more nimble he is and therefore less likely to hurt himself on landing.
- The environment variable HUNT is checked to get the player name. If you don't have this variable set, **hunt** will ask you what name you want to play under. If it is set, you may also set up a single character keyboard map, but then you have to enumerate the options; for example,


```
setenv HUNT "name=Sneaky,mapkey=z0FfGg1f2g3F4G"
```

 sets the player name to Sneaky, and the maps z to o, F to f, G to g, 1 to f, 2 to g, 3 to F, and 4 to G. The **mapkey** option must be last.
- It's a boring game if you're the only one playing.

FILES

`/usr/games/lib/hunt.driver` game coordinator

NAME

mastermind – Mastermind guessing game

SYNOPSIS

/usr/games/mastermind

DESCRIPTION

This program plays the game of "mastermind". The playing field is a number of slots, in which a number of colored pegs can be placed. The object of the game is to guess what color peg is in each slot. You and the program take turns trying to guess each other's configuration.

Before play begins, **mastermind** asks you whether you want instructions. Respond by typing a "y" for yes or an "n" for no. Following this, you have a chance to decide how many slots and how many colors you want to use. When you enter a guess, type the names of the colors, separated by spaces. When the program makes a guess, respond with two digits separated by spaces.

A guess consists of a possible sequence of colored pegs. The guesser's opponent answers with two numbers: the number of pegs in the guess that exactly match the corresponding pegs in the configuration, and the number of pegs in the guess that match in color but not in position. For example, suppose you are playing with five slots, and the following situation occurs:

```
my configuration: red  red  yellow blue  brown
your guess:      blue red  green red  red
```

The two numbers would then be 1 and 2. The 1 applies because both you and the program have a red peg in the second slot. In addition, your blue matches the program's blue, though the position is wrong, and one of your reds matches the program's red in the first slot. Only two of your reds match because the program only has two reds in its configuration.

Any time it is your turn to enter a guess, you can ask the program what happened by typing "review" instead of your guess. You get one point for each guess that the program has to make, and it gets one point for each guess that you have to make.

NAME

maze – generate a maze

SYNOPSIS

/usr/games/maze [*seed* [**-n**]]

DESCRIPTION

maze generates a conventional rectangular labyrinth. To set the random number generator to a particular seed value, supply an integer *seed* as the argument. Invoking **maze** with a given seed always produces the same maze.

If you provide a seed value, **maze** also shows the solution to the labyrinth. To suppress this effect, use the **-n** option.

BUGS

Assumes hardcopy output when displaying the solution.

NAME

melt – “melt” the screen

SYNOPSIS

/usr/games/melt

DESCRIPTION

This program creates a melted screen effect.

NAME

mille – play Mille Bournes

SYNOPSIS

/usr/games/mille [*file*]

DESCRIPTION

mille plays a two-handed game reminiscent of the Parker Brother's game of Mille Bournes with you. The rules are described below. If a file name is given on the command line, the game saved in that file is started.

When a game is started up, the bottom of the score window will contain a list of commands. They are:

- P** Pick a card from the deck. This card is placed in the "P" slot in your hand.
- D** Discard a card from your hand. To indicate which card, type the number of the card in the hand (or "P" for the just-picked card) followed by a <RETURN> or <SPACE>. The <RETURN> or <SPACE> is required to allow recovery from typos which can be very expensive, like discarding safeties.
- U** Use a card. The card is again indicated by its number, followed by a <RETURN> or <SPACE>.
- O** Toggle ordering the hand. By default off, if turned on it will sort the cards in your hand appropriately. This is not recommended for the impatient on slow terminals.
- Q** Quit the game. This will ask for confirmation, just to be sure. Hitting <DELETE> (or <RUBOUT>) is equivalent.
- S** Save the game in a file. If the game was started from a file, you will be given an opportunity to save it on the same file. If you don't wish to, or you did not start from a file, you will be asked for the file name. If you type a <RETURN> without a name, the save will be terminated and the game resumed.
- R** Redraw the screen from scratch. The command CTRL/L will also work.
- W** Toggle window type. This switches the score window between the startup window (with all the command names) and the end-of-game window. Using the end-of-game window saves time by eliminating the switch at the end of the game to show the final score. Recommended for hackers and other miscreants.

If you make a mistake, an error message will be printed on the last line of the score window, and a bell will beep.

At the end of each hand or game, you will be asked if you wish to play another. If not, it will ask you if you want to save the game. If you do, and the save is unsuccessful, play will be resumed as if you had said you wanted to play another hand/game. This allows you to use the "S" command to reattempt the save.

CARDS

Here is some useful information. The number in parentheses after the card name is the number of that card in the deck:

Hazard	Repair	Safety
Out of Gas (2)	Gasoline (6)	Extra Tank (1)
Flat Tire (2)	Spare Tire (6)	Puncture Proof (1)
Accident (2)	Repairs (6)	Driving Ace (1)
Stop (4)	Go (14)	Right of Way (1)
Speed Limit (3)	End of Limit (6)	

25 – (10), 50 – (10), 75 – (10), 100 – (12), 200 – (4)

RULES

Object: The point of this game is to get a total of 5000 points in several hands. Each hand is a race to put down exactly 700 miles before your opponent does. Beyond the points gained by putting down milestones, there are several other ways of making points.

Overview: The game is played with a deck of 101 cards. **Distance** cards represent a number of miles traveled. They come in denominations of 25, 50, 75, 100, and 200. When one is played, it adds that many miles to the player's trip so far this hand. **Hazard** cards are used to prevent your opponent from putting down Distance cards. They can only be played if your opponent has a Go card on top of the Battle pile. The cards are **Out of Gas**, **Accident**, **Flat Tire**, **Speed Limit**, and **Stop**. **Remedy** cards fix problems caused by Hazard cards played on you by your opponent. The cards are **Gasoline**, **Repairs**, **Spare Tire**, **End of Limit**, and **Go**. **Safety** cards prevent your opponent from putting specific Hazard cards on you in the first place. They are **Extra Tank**, **Driving Ace**, **Puncture Proof**, and **Right of Way**, and there are only one of each in the deck.

Board Layout: The board is split into several areas. From top to bottom, they are: **SAFETY AREA** (unlabeled): This is where the safeties will be placed as they are played. **HAND:** These are the cards in your hand. **BATTLE:** This is the Battle pile. All the Hazard and Remedy Cards are played here, except the **Speed Limit** and **End of Limit** cards. Only the top card is displayed, as it is the only effective one. **SPEED:** The Speed pile. The **Speed Limit** and **End of Limit** cards are played here to control the speed at which the player is allowed to put down miles. **MILEAGE:** Miles are placed here. The total of the numbers shown here is the distance traveled so far.

Play: The first pick alternates between the two players. Each turn usually starts with a pick from the deck. The player then plays a card, or if this is not possible or desirable, discards one. Normally, a play or discard of a single card constitutes a turn. If the card played is a safety, however, the same player takes another turn immediately.

This repeats until one of the players reaches 700 points or the deck runs out. If someone reaches 700, they have the option of going for an *Extension*, which means that

the play continues until someone reaches 1000 miles.

Hazard and Remedy Cards: Hazard Cards are played on your opponent's Battle and Speed piles. Remedy Cards are used for undoing the effects of your opponent's nastiness.

Go (Green Light) must be the top card on your Battle pile for you to play any mileage, unless you have played the **Right of Way** card (see below).

Stop is played on your opponent's **Go** card to prevent them from playing mileage until they play a **Go** card.

Speed Limit is played on your opponent's Speed pile. Until they play an **End of Limit** they can only play 25 or 50 mile cards, presuming their **Go** card allows them to do even that.

End of Limit is played on your Speed pile to nullify a **Speed Limit** played by your opponent.

Out of Gas is played on your opponent's *Go* card. They must then play a *Gasoline* card, and then a *Go* card before they can play any more mileage.

Flat Tire is played on your opponent's *Go* card. They must then play a *Spare Tire* card, and then a *Go* card before they can play any more mileage.

Accident is played on your opponent's *Go* card. They must then play a *Repairs* card, and then a *Go* card before they can play any more mileage.

Safety Cards: Safety cards prevent your opponent from playing the corresponding Hazard cards on you for the rest of the hand. It cancels an attack in progress, and *always entitles the player to an extra turn.*

Right of Way prevents your opponent from playing both *Stop* and *Speed Limit* cards on you. It also acts as a permanent *Go* card for the rest of the hand, so you can play mileage as long as there is not a Hazard card on top of your Battle pile. In this case only, your opponent can play Hazard cards directly on a Remedy card other than a *Go* card.

Extra Tank When played, your opponent cannot play an **Out of Gas** on your Battle Pile.

Puncture Proof When played, your opponent cannot play a **Flat Tire** on your Battle Pile.

Driving Ace When played, your opponent cannot play an **Accident** on your Battle Pile.

Distance Cards: Distance cards are played when you have a *Go* card on your Battle pile, or a *Right of Way* in your Safety area and are not stopped by a Hazard Card. They can be played in any combination that totals exactly 700 miles, except that *you cannot play more than two 200 mile cards in one hand.* A hand ends whenever one player gets exactly 700 miles or the deck runs out. In that case, play continues until neither someone reaches 700, or neither player can use any cards in their hand. If the trip is completed after the deck runs out, this is called **Delayed Action.**

Coup Fourré: This is a French fencing term for a counter-thrust move as part of a parry to an opponents attack. In Mille Bourne, it is used as follows: If an opponent plays a Hazard card, and you have the corresponding Safety in your hand, you play it immediately, even *before* you draw. This immediately removes the Hazard card from

your Battle pile, and protects you from that card for the rest of the game. This gives you more points (see "Scoring" below).

Scoring: Scores are totaled at the end of each hand, whether or not anyone completed the trip. The terms used in the Score window have the following meanings:

Milestones Played : Each player scores as many miles as they played before the trip ended.

Each Safety : 100 points for each safety in the Safety area.

All 4 Safeties : 300 points if all four safeties are played.

Each Coup Fourré : 300 points for each Coup Fourré accomplished.

The following bonus scores can apply only to the winning player.

Trip Completed : 400 points bonus for completing the trip to 700 or 1000.

Safe Trip : 300 points bonus for completing the trip without using any 200 mile cards.

Delayed Action : 300 points bonus for finishing after the deck was exhausted.

Extension : 200 points bonus for completing a 1000 mile trip.

Shut-Out : 500 points bonus for completing the trip before your opponent played any mileage cards.

Running totals are also kept for the current score for each player for the hand (**Hand Total**), the game (**Overall Total**), and number of games won (**Games**).

NAME

monop – Monopoly game

SYNOPSIS

/usr/games/monop

DESCRIPTION

monop is reminiscent of the Parker Brothers game Monopoly for 1 to 9 players. The program assumes that the rules of Monopoly are known. It follows the standard rules, with the exception that if a property goes up for auction and there are only two solvent players, no auction is held and the property remains unowned.

The game, in effect, lends the player money, so it is possible to buy something which you cannot afford. However, as soon as a person goes into debt, he must "fix the problem", (i.e., make himself solvent) before play can continue. If this is not possible, the player's property reverts to his debtee (either a player or the bank). A player can resign at any time to any person or the bank, which puts the property back on the board unowned.

Any time that the expected response to a question is a string, (for instance, a name, place or person) you can type a question mark (?) to get a list of valid answers. It is not possible to input a negative number, nor is it ever necessary.

COMMANDS

- quit** Quit the game. Asks for confirmation.
- print** Print out the current board. The columns have the following meanings:
- Name** The first ten characters of the name of the square
 - Own** The player number of the owner of the property
 - Price** The cost of the property (if any)
 - Mg** This field has an asterisk (*) in it if the property is mortgaged
 - #** If the property is a Utility or Railroad, this is the number of such owned by the owner. If the property is land, this is the number of houses on it.
 - Rent** Current rent on the property. If it is not owned, there is no rent.
- where** Tell where all the players are. An asterisk (*) indicates the current player.
- own holdings** List your own holdings (money, "get out of jail free" cards, and property).
- holdings** Look at anyone's holdings. The program will ask you whose holdings you wish to look at. When you are finished, type "done".
- shell** Escape to a shell. When the shell dies, the program continues where you left off.
- mortgage** Display a list of mortgageable property. Asks which you wish to mortgage.
- unmortgage** Unmortgage mortgaged property.

- buy** Display a list of monopolies on which you can buy houses. If there is more than one, asks you which you want to buy for, and then asks you how many for each piece of property, giving the current amount in parentheses after the property name. If you build in an unbalanced manner (a disparity of more than one house within the same monopoly), asks you to re-enter values.
- sell** Display a list of monopolies from which you can sell houses. It operates in a manner analogous to **buy**.
- card** Use a "get out of jail free" card. Informs you if you're not in jail, or you don't have such a card.
- pay** Pay \$50 to get out of jail, whence you are put on the Just Visiting space.
- trade** Trade with another player. Asks you whom you wish to trade with, and then asks you what each wishes to give up. You can get a summary at the end, and, in all cases, it asks for confirmation of the trade before completing it.
- resign** Resign to another player or the bank. If you resign to the bank, all property reverts to its virgin state, and "get out of jail free" cards revert to the deck.
- save** Save the current game in a file for later play. You can continue play later on either by giving the save file's name as an argument to the **monop** command, or by using the **restore** command (see below). Asks you which file you wish to save to; if the file exists, asks you to confirm that you wish to overwrite it.
- restore** Read in a previously saved game from a file. Leaves the file intact.
- roll** Roll the dice and move forward to your new location. If you simply press <RETURN> without entering a command, a roll is performed.

FILES

/usr/games/lib/cards.pck Chance and Community Chest cards

BUGS

No command can be given an argument instead of a response to a query.

NAME

moo – guessing game

SYNOPSIS

/usr/games/moo

DESCRIPTION

moo is a guessing game imported from England. The computer picks a four-digit number which you try to guess; **moo** scores you on each guess. A “cow” is a correct digit in an incorrect position. A “bull” is a correct digit in a correct position. The game continues until you guess the number (a score of four bulls).

NUMBER(6)

BSD

NUMBER(6)

NAME

number – convert Arabic numerals to English

SYNOPSIS

/usr/games/number

DESCRIPTION

number copies the standard input to the standard output, changing each decimal number to a fully spelled-out version.

NAME

primes – print prime numbers

SYNOPSIS

/usr/games/primes [*number*]

DESCRIPTION

primes displays the prime numbers equal to or greater than the input *number*. If you do not provide an argument, **primes** reads a line from the standard input. To exit the program, type an interrupt.

NAME

puzzle – puzzle game

SYNOPSIS

/usr/games/puzzle

DESCRIPTION

puzzle lets you play the familiar 15-tile game. To obtain online help, position the cursor over the puzzle border and press the HELP key.

COMMANDS

Each of the mouse buttons has a particular function:

Left Use to select items from the menu, and to move individual tiles.

Middle Back up one move.

Right Move one step forward in solving the puzzle.

SEE ALSO

scramble(6)

NAME

quiz – test your knowledge

SYNOPSIS

/usr/games/quiz [**-i** *file*] [**-t**] [*category1 category2*]

DESCRIPTION

quiz gives associative knowledge tests on various subjects. It asks items chosen from *category1* and expects answers from *category2*. If no categories are specified, **quiz** gives instructions and lists the available categories.

quiz tells a correct answer whenever you type a bare newline. At the end of input, upon interrupt, or when questions run out, **quiz** reports a score and terminates.

The lines of the index file have the following syntax:

```
line      = category newline | category ':' line
category = alternate | category '[' alternate
alternate = empty | alternate primary
primary   = character | '[' category '[' | option
option    = '{' category '{'
```

The first category on each line of an index file names an information file. The remaining categories specify the order and contents of the data in each line of the information file. Information files have the same syntax. Use a backslash (\) as with **sh**(1) to quote syntactically significant characters or to insert transparent newlines into a line. When either a question or its answer is empty, **quiz** will refrain from asking it.

OPTIONS

-t Enable tutorial mode, where missed questions are repeated later, and material is gradually introduced as you learn.

-i file Substitute the named file for the default index file.

FILES

/usr/games/quiz.k/*

BUGS

The construct 'a|ab' doesn't work in an information file. Use 'a{b}'.

NAME

rain -- animated raindrops display

SYNOPSIS

/usr/games/rain

DESCRIPTION

rain simulates the pitter-patter of raindrops falling on your tty. It looks best at 9600 baud or faster. As with all programs that use **termcap**(3), the **TERM** environment variable must be set (and exported) to the type of the terminal being used.

rain can only be halted with an interrupt.

FILES

/etc/termcap

NAME

random – random number generator

SYNOPSIS

`/usr/games/random` [*numbers*] [*characters*]

DESCRIPTION

random is a random number generator that accepts input of numbers or characters. There is about a 50% probability that the program will match and output the lines that you use as input. You can also perform a particular task (e.g., combining lines in a file, listing a directory, etc.) and then use **random** in a pipeline to produce a random sampling of the output. The **random** command is a likely candidate for use in other games that require the use of a random number generator.

EXAMPLE

The following command line lists a random percentage (about 50%) of the contents of **mydir**:

```
% ls mydir | random
```

NAME

revscr – reverse screen

SYNOPSIS

/usr/games/revscr

DESCRIPTION

revscr reflects the display image twice, first about the horizontal axis and then about the vertical axis. The effect is to turn the image upside down. Press any key to restore the display to its normal state.

OPTIONS

-short Reflect about the horizontal axis only. Useful on machines which perform the second reflection slowly.

NAME

robots – fight off villainous robots

SYNOPSIS

`/usr/games/robots [ -sjta ] [ scorefile ]`

DESCRIPTION

robots pits you against evil robots who are trying to kill you (which is why they are evil). Fortunately for you, although they are evil they are not very bright. The robots have a habit of bumping into each other, thus turning themselves into junkpiles. In order to survive, you must get them to kill each other off, since you have no offensive weaponry. You are endowed with one defensive measure: a teleportation device.

When two robots run into each other or a junk pile, they die. If a robot runs into you, you die. When a robot dies, you get 10 points, and when all the robots die, you start on the next field. This keeps up until they finally get you.

Robots are represented on the screen by plus signs (+), the junk heaps resulting from their collisions by asterisks (*), and you (the good guy) by an at sign (@).

Only five scores are allowed per user on the score file. If you make it into the score file, you will be shown the list at the end of the game. If an alternate score file is specified, that will be used instead of the standard file for scores.

COMMANDS

All commands can be preceded by a count. For all commands except **w**, the program saves you from typos by stopping you short of being eaten. However, with **w** you take the risk of dying by miscalculation.

- h** Move one square left
- l** Move one square right
- k** Move one square up
- j** Move one square down
- y** Move one square up and left
- u** Move one square up and right
- b** Move one square down and left
- n** Move one square down and right
- . or <SPACE>** Do nothing for one turn
- HJKLBNYU** Run as far as possible in the given direction
- >** Do nothing for as long as possible
- t** Teleport to a random location
- w** Wait until they all die (or you do). If you use the **w** command and survive to the next level, you will get a bonus of 10% for each robot which died after you decided to wait. If you die, however, you get nothing.

q Quit
^L Redraw the screen

OPTIONS

- s** Don't play, just show the score file
- j** Jump, *i.e.*, when you run, don't show any intermediate positions; only show things at the end. This is useful on slow terminals.
- t** Teleport automatically when you have no other option. This is a little disconcerting until you get used to it, and then it is very nice.
- a** Advance into the higher levels directly, skipping the lower, easier levels. You receive a 600-point bonus for successful completion of the first field.

FILES

/usr/games/lib/robots_roll the score file

NAME

sail – multi-user wooden ships and iron men

SYNOPSIS

sail [**-s** [**-l**]] [**-x**] [**-b**] [*num*]

DESCRIPTION

sail is a computer version of Avalon Hill's game of fighting sail originally developed by S. Craig Taylor.

Players of **sail** take command of an old fashioned Man of War and fight other players or the computer. They may re-enact one of the many historical sea battles recorded in the game, or they can choose a fictional battle.

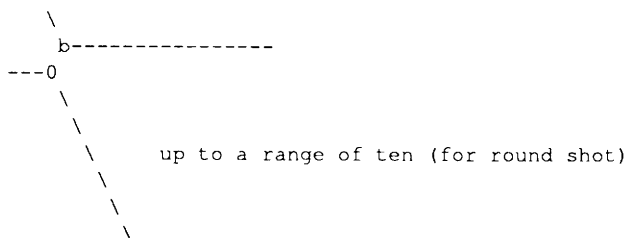
As a sea captain in the **sail** Navy, the player has complete control over the workings of his ship. He must order every maneuver, change the set of his sails, and judge the right moment to let loose the terrible destruction of his broadsides. In addition to fighting the enemy, he must harness the powers of the wind and sea to make them work for him. The outcome of many battles during the age of sail was decided by the ability of one captain to hold the "weather gauge."

OPTIONS

- s** Print the names and ships of the top ten sailors.
- l** Show the login name. Only effective with **-s**.
- x** Play the first available ship instead of prompting for a choice.
- b** No bells.

HISTORICAL INFO

Old Square Riggers were very maneuverable ships capable of intricate sailing. Their only disadvantage was an inability to sail very close to the wind. The design of a wooden ship allowed only for the guns to bear to the left and right sides. A few guns of small aspect (usually 6 or 9 pounders) could point forward, but their effect was small compared to a 68 gun broadside of 24 or 32 pounders. The guns bear approximately like so:



An interesting phenomenon occurred when a broadside was fired down the length of an enemy ship. The shot tended to bounce along the deck and did several times more

damage. This phenomenon was called a rake. Because the bows of a ship are very strong and present a smaller target than the stern, a stern rake (firing from the stern to the bow) causes more damage than a bow rake.

```

      b
00    ----  Stern rake!
      a

```

Most ships were equipped with carronades, which were very large, close range cannons. American ships from the revolution until the War of 1812 were almost entirely armed with carronades. The period of history covered in *sail* runs approximately from the 1770's until the end of Napoleonic France in 1815.

Fighting ships came in several sizes classed by armament. The mainstays of any fleet were its "Ships of the Line", or "Line of Battle Ships". They were so named because these ships fought together in great lines. They were close enough for mutual support, yet every ship could fire both its broadsides. We get the modern words "ocean liner," or "liner," and "battleship" from "ship of the line." The most common size was the the 74 gun two decked ship of the line. The two gun decks usually mounted 18 and 24 pounder guns.

The pride of the fleet were the first rates. These were huge three decked ships of the line mounting 80 to 136 guns. The guns in the three tiers were usually 18, 24, and 32 pounders in that order from top to bottom.

Various other ships came next. They were almost all "razees," or ships of the line with one deck sawed off. They mounted 40-64 guns and were a poor cross between a frigate and a line of battle ship. They neither had the speed of the former nor the firepower of the latter.

Next came the "eyes of the fleet." Frigates came in many sizes mounting anywhere from 32 to 44 guns. They were very handy vessels. They could outsail anything bigger and outshoot anything smaller. Frigates didn't fight in lines of battle as the much bigger 74's did. Instead, they harassed the enemy's rear or captured crippled ships. They were much more useful in missions away from the fleet, such as cutting out expeditions or boat actions. They could hit hard and get away fast.

Lastly, there were the corvettes, sloops, and brigs. These were smaller ships mounting typically fewer than 20 guns. A corvette was only slightly smaller than a frigate, so one might have up to 30 guns. Sloops were used for carrying dispatches or passengers. Brigs were something you built for land-locked lakes.

SAIL PARTICULARS

Ships in *sail* are represented by two characters. One character represents the bow of the ship, and the other represents the stern. Ships have nationalities and numbers. The first ship of a nationality is number 0, the second number 1, etc. Therefore, the first British ship in a game would be printed as "b0". The second Brit would be "b1", and the fifth Don would be "s4".

Ships can set normal sails, called Battle Sails, or bend on extra canvas called Full Sails. A ship under full sail is a beautiful sight indeed, and it can move much faster than a ship under Battle Sails. The only trouble is, with full sails set, there is so much tension on sail and rigging that a well aimed round shot can burst a sail into ribbons where it would only cause a little hole in a loose sail. For this reason, rigging damage is doubled on a ship with full sails set.

A ship with full sails set has a capital letter for its nationality. For example, a French ship, normally "f0", with full sails set would be printed as "F0".

When a ship is battered into a listing hulk, the last man aboard "strikes the colors." This ceremony is the ship's formal surrender. The nationality character of a surrendered ship is printed as "!". Thus, the French ship of the last example would soon be "10".

A ship has a random chance of catching fire or sinking when it reaches the stage of listing hulk. A sinking ship has a tilde (~) printed for its nationality, and a ship on fire and about to explode has a pound sign (#) printed.

Captured ships become the nationality of the prize crew. Therefore, if an American ship captures a British ship, the British ship will have an "a" printed for its nationality. In addition, the ship number is changed to "&","", "(","),","*", or "+" depending upon the original number, be it 0,1,2,3,4, or 5. E.g., the "b0" captured by an American becomes the "a&". The "s4" captured by the French becomes the "f*".

MOVEMENT

Movement is the most confusing part of **sail** to many. Ships can head in 8 directions:

			0	0	0		
b	b	b0	b	b	b	0b	b
0	0						0

The stern of a ship moves when it turns. The bow remains stationary. Ships can always turn, regardless of the wind (unless they are becalmed). All ships drift when they lose headway. If a ship doesn't move forward at all for two turns, it will begin to drift. If a ship has begun to drift, then it must move forward before it turns, if it plans to do more than make a right or left turn, which is always possible.

Movement commands to **sail** are a string of forward moves and turns. An example is "13". It will turn a ship left and then move it ahead 3 spaces. In the drawing above, the "b0" made 7 successive left turns. When **sail** prompts you for a move, it prints three characters of import, e.g.,

```
move (7, 4):
```

The first number is the maximum number of moves you can make, including turns. The second number is the maximum number of turns you can make. Between the numbers is sometimes printed a single quote ('). If the quote is present, it means that your ship has been drifting, and you must move ahead to regain headway before you turn (see note above). Some of the possible moves for the example above are as

follows:

```

move (7, 4): 7
move (7, 4): 1
move (7, 4): d          /* drift, or do nothing */
move (7, 4): 6r
move (7, 4): 5r1
move (7, 4): 11r1r2

```

Because square riggers performed so poorly sailing into the wind, if at any point in a movement command you turn into the wind, the movement stops there:

```

move (7, 4): 1114
Movement Error;
Helm: 111

```

Moreover, whenever you make a turn, your movement allowance drops to the lesser of a) what's left and b) what you would have at the new attitude. In short, if you turn closer to the wind, you most likely won't be able to sail the full allowance printed in the "move" prompt.

Old sailing captains had to keep an eye constantly on the wind. Captains in *sail* are no different. A ship's ability to move depends on its attitude to the wind. The best angle possible is to have the wind off your quarter, that is, just off the stern. The direction rose on the side of the screen gives the possible movements for your ship at all positions to the wind. Battle sail speeds are given first, and full sail speeds are given in parenthesis.

```

  0 1(2)
  \ /
 - ^ - 3(6)
  / \
  | 4(7)
  3(6)

```

Pretend the bow of your ship (the "^") is pointing upward and the wind is blowing from the bottom to the top of the page. The numbers at the bottom "3(6)" will be your speed under battle or full sails in such a situation. If the wind is off your quarter, then you can move "4(7)". If the wind is off your beam, "3(6)". If the wind is off your bow, then you can only move "1(2)". Facing into the wind, you can't move at all. Ships facing into the wind were said to be "in irons".

WINDSPEED AND DIRECTION

The windspeed and direction is displayed as a little weather vane on the side of the screen. The number in the middle of the vane indicates the wind speed, and the + to - indicates the wind direction. The wind blows from the + sign (high pressure) to the - sign (low pressure). E.g.,

|
3
+

The wind speeds are 0 = becalmed, 1 = light breeze, 2 = moderate breeze, 3 = fresh breeze, 4 = strong breeze, 5 = gale, 6 = full gale, 7 = hurricane. If a hurricane shows up, all ships are destroyed.

GRAPPLING AND FOULING

If two ships collide, they run the risk of becoming tangled together. This is called "fouling." Fouled ships are stuck together, and neither can move. They can unfoul each other if they want to. Boarding parties can only be sent across to ships when the antagonists are either fouled or grappled.

Ships can grapple each other by throwing grapnels into the rigging of the other.

The number of fouls and grapples you have are displayed on the upper right of the screen.

BOARDING

Boarding was a very costly venture in terms of human life. Boarding parties may be formed in **sail** to either board an enemy ship or to defend your own ship against attack. Men organized as Defensive Boarding Parties fight twice as hard to save their ship as men left unorganized.

The boarding strength of a crew depends upon its quality and upon the number of men sent.

CREW QUALITY

The British seaman was world renowned for his sailing abilities. American sailors, however, were actually the best seamen in the world. Because the American Navy offered twice the wages of the Royal Navy, British seamen who liked the sea defected to America by the thousands.

In **sail**, crew quality is quantized into 5 energy levels. "Elite" crews can outshoot and outfight all other sailors. "Crack" crews are next. "Mundane" crews are average, and "Green" and "Mutinous" crews are below average. A good rule of thumb is that "Crack" or "Elite" crews get one extra hit per broadside compared to "Mundane" crews. Don't expect too much from "Green" crews.

BROADSIDES

Your two broadsides may be loaded with four kinds of shot: grape, chain, round, and double. You have guns and carronades in both the port and starboard batteries. Carronades only have a range of two, so you have to get in close to be able to fire them. You have the choice of firing at the hull or rigging of another ship. If the range of the ship is greater than 6, then you may only shoot at the rigging.

The types of shot and their advantages are:

Round	Range of 10. Good for hull or rigging hits.
Double	Range of 1. Extra good for hull or rigging hits. Double takes two turns to load.

Chain Range of 3. Excellent for tearing down rigging. Cannot damage hull or guns, though.

Grape Range of 1. Sometimes devastating against enemy crews.

On the side of the screen is displayed some vital information about your ship:

Load D! R!
Hull 9
Crew 4 4 2
Guns 4 4
Carr 2 2
Rigg 5 5 5

"Load" shows what your port (left) and starboard (right) broadsides are loaded with. A "!" after the type of shot indicates that it is an initial broadside. Initial broadsides were loaded with care before battle and before the decks ran red with blood. As a consequence, initial broadsides are a little more effective than broadsides loaded later. A "*" after the type of shot indicates that the gun crews are still loading it, and you cannot fire yet. "Hull" shows how much hull you have left. "Crew" shows your three sections of crew. As your crew dies off, your ability to fire decreases. "Guns" and "Carr" show your port and starboard guns. As you lose guns, your ability to fire decreases. "Rigg" shows how much rigging you have on your 3 or 4 masts. As rigging is shot away, you lose mobility.

EFFECTIVENESS OF FIRE

It is very dramatic when a ship fires its thunderous broadsides, but the mere opportunity to fire them does not guarantee any hits. Many factors influence the destructive force of a broadside. First of all, and the chief factor, is distance. It is harder to hit a ship at range ten than it is to hit one sloshing alongside. Next is raking. Raking fire, as mentioned before, can sometimes dismast a ship at range ten. Next, crew size and quality affects the damage done by a broadside. The number of guns firing also bears on the point, so to speak. Lastly, weather affects the accuracy of a broadside. If the seas are high (5 or 6), then the lower gunports of ships of the line can't even be opened to run out the guns. This gives frigates and other flush decked vessels an advantage in a storm. The scenario *Pellew vs. The Droits de L'Homme* takes advantage of this peculiar circumstance.

REPAIRS

Repairs may be made to your Hull, Guns, and Rigging at the slow rate of two points per three turns. The message "Repairs Completed" will be printed if no more repairs can be made.

PECULIARITIES OF COMPUTER SHIPS

Computer ships in *sail* follow all the rules above with a few exceptions. Computer ships never repair damage. If they did, the players could never beat them. They play well enough as it is. As a consolation, the computer ships can fire double shot every turn. That fluke is a good reason to keep your distance. The driver figures out the moves of the computer ships. It computes them with a typical distance function and a depth-first search to find the maximum "score."

COMMANDS

Commands are given to **sail** by typing a single character. You will then be prompted for further input.

f	Fire broadsides if they bear
l	Reload
L	Unload broadsides (to change ammo)
m	Move
i	Print the closest ship
I	Print all ships
F	Find a particular ship or ships (e.g. "a?" for all Americans)
s	Send a message around the fleet
b	Attempt to board an enemy ship
B	Recall boarding parties
c	Change set of sail
r	Repair
u	Attempt to unfoul
g	Grapple/ungrapple
v	Print version number of game
CTRL/L	Redraw screen
Q	Quit
C	Center your ship in the window
U	Move window up
D, N	Move window down
H	Move window left
J	Move window right
S	Toggle window to follow your ship or stay where it is

SCENARIOS

Here is a summary of the scenarios in **sail**:

Ranger vs. Drake:

Wind from the N, blowing a fresh breeze.

(a) Ranger	19 gun Sloop (crack crew) (7 pts)
(b) Drake	17 gun Sloop (crack crew) (6 pts)

The Battle of Flamborough Head:

Wind from the S, blowing a fresh breeze.

This is John Paul Jones' first famous battle. Aboard the Bonhomme Richard, he was able to overcome the Serapis's greater firepower by quickly boarding her.

- (a) Bonhomme Rich 42 gun Corvette (crack crew) (11 pts)
- (b) Serapis 44 gun Frigate (crack crew) (12 pts)

Arbuthnot and Des Touches:

Wind from the N, blowing a gale.

- (b) America 64 gun Ship of the Line (crack crew) (20 pts)
- (b) Befford 74 gun Ship of the Line (crack crew) (26 pts)
- (b) Adamant 50 gun Ship of the Line (crack crew) (17 pts)
- (b) London 98 gun 3 Decker SOL (crack crew) (28 pts)
- (b) Royal Oak 74 gun Ship of the Line (crack crew) (26 pts)
- (f) Neptune 74 gun Ship of the Line (average crew) (24 pts)
- (f) Duc Bougogne 80 gun 3 Decker SOL (average crew) (27 pts)
- (f) Conquerant 74 gun Ship of the Line (average crew) (24 pts)
- (f) Provence 64 gun Ship of the Line (average crew) (18 pts)
- (f) Romulus 44 gun Ship of the Line (average crew) (10 pts)

Suffren and Hughes:

Wind from the S, blowing a fresh breeze.

- (b) Monmouth 74 gun Ship of the Line (average crew) (24 pts)
- (b) Hero 74 gun Ship of the Line (crack crew) (26 pts)
- (b) Isis 50 gun Ship of the Line (crack crew) (17 pts)
- (b) Superb 74 gun Ship of the Line (crack crew) (27 pts)
- (b) Burford 74 gun Ship of the Line (average crew) (24 pts)
- (f) Flambrant 50 gun Ship of the Line (average crew) (14 pts)
- (f) Annibal 74 gun Ship of the Line (average crew) (24 pts)
- (f) Severe 64 gun Ship of the Line (average crew) (18 pts)
- (f) Brilliant 80 gun Ship of the Line (crack crew) (31 pts)
- (f) Sphinx 80 gun Ship of the Line (average crew) (27 pts)

Nymphe vs. Cleopatre:

Wind from the S, blowing a fresh breeze.

- (b) Nymphe 36 gun Frigate (crack crew) (11 pts)
- (f) Cleopatre 36 gun Frigate (average crew) (10 pts)

Mars vs. Hercule:

Wind from the S, blowing a fresh breeze.

- (b) Mars 74 gun Ship of the Line (crack crew) (26 pts)
- (f) Hercule 74 gun Ship of the Line (average crew) (23 pts)

Ambuscade vs. Baionnaise:

Wind from the N, blowing a fresh breeze.

- (b) Ambuscade 32 gun Frigate (average crew) (9 pts)
- (f) Baionnais 24 gun Corvette (average crew) (9 pts)

Constellation vs. Insurgent:

Wind from the S, blowing a gale.

- (a) Constellation 38 gun Corvette (elite crew) (17 pts)
- (f) Insurgent 36 gun Corvette (average crew) (11 pts)

Constellation vs. Vengeance:

Wind from the S, blowing a fresh breeze.

- (a) Constellation 38 gun Corvette (elite crew) (17 pts)
- (f) Vengeance 40 gun Frigate (average crew) (15 pts)

The Battle of Lissa:

Wind from the S, blowing a fresh breeze.

- (b) Amphion 32 gun Frigate (elite crew) (13 pts)
- (b) Active 38 gun Frigate (elite crew) (18 pts)
- (b) Volage 22 gun Frigate (elite crew) (11 pts)
- (b) Cerberus 32 gun Frigate (elite crew) (13 pts)
- (f) Favorite 40 gun Frigate (average crew) (15 pts)
- (f) Flore 40 gun Frigate (average crew) (15 pts)
- (f) Danae 40 gun Frigate (crack crew) (17 pts)
- (f) Bellona 32 gun Frigate (green crew) (9 pts)
- (f) Corona 40 gun Frigate (green crew) (12 pts)
- (f) Carolina 32 gun Frigate (green crew) (7 pts)

Constitution vs. Guerriere:

Wind from the SW, blowing a gale.

- (a) Constitution 44 gun Corvette (elite crew) (24 pts)
- (b) Guerriere 38 gun Frigate (crack crew) (15 pts)

United States vs. Macedonian:

Wind from the S, blowing a fresh breeze.

- (a) United States 44 gun Frigate (elite crew) (24 pts)
- (b) Macedonian 38 gun Frigate (crack crew) (16 pts)

Constitution vs. Java:

Wind from the S, blowing a fresh breeze.

- (a) Constitution 44 gun Corvette (elite crew) (24 pts)
- (b) Java 38 gun Corvette (crack crew) (19 pts)

Chesapeake vs. Shannon:

Wind from the S, blowing a fresh breeze.

- (a) Chesapeake 38 gun Frigate (average crew) (14 pts)

- (b) Shannon 38 gun Frigate (elite crew) (17 pts)

The Battle of Lake Erie:

Wind from the S, blowing a light breeze.

- (a) Lawrence 20 gun Sloop (crack crew) (9 pts)
 (a) Niagara 20 gun Sloop (elite crew) (12 pts)
 (b) Lady Prevost 13 gun Brig (crack crew) (5 pts)
 (b) Detroit 19 gun Sloop (crack crew) (7 pts)
 (b) Q. Charlotte 17 gun Sloop (crack crew) (6 pts)

Wasp vs. Reindeer:

Wind from the S, blowing a light breeze.

- (a) Wasp 20 gun Sloop (elite crew) (12 pts)
 (b) Reindeer 18 gun Sloop (elite crew) (9 pts)

Constitution vs. Cyane and Levant:

Wind from the S, blowing a moderate breeze.

- (a) Constitution 44 gun Corvette (elite crew) (24 pts)
 (b) Cyane 24 gun Sloop (crack crew) (11 pts)
 (b) Levant 20 gun Sloop (crack crew) (10 pts)

Pellew vs. Droits de L'Homme:

Wind from the N, blowing a gale.

- (b) Indefatigable 44 gun Frigate (elite crew) (14 pts)
 (b) Amazon 36 gun Frigate (crack crew) (14 pts)
 (f) Droits L'Hom 74 gun Ship of the Line (average crew) (24 pts)

Algeciras:

Wind from the SW, blowing a moderate breeze.

- (b) Caesar 80 gun Ship of the Line (crack crew) (31 pts)
 (b) Pompee 74 gun Ship of the Line (crack crew) (27 pts)
 (b) Spencer 74 gun Ship of the Line (crack crew) (26 pts)
 (b) Hannibal 98 gun 3 Decker SOL (crack crew) (28 pts)
 (s) Real-Carlos 112 gun 3 Decker SOL (green crew) (27 pts)
 (s) San Fernando 96 gun 3 Decker SOL (green crew) (24 pts)
 (s) Argonauta 80 gun Ship of the Line (green crew) (23 pts)
 (s) San Augustine 74 gun Ship of the Line (green crew) (20 pts)
 (f) Indomptable 80 gun Ship of the Line (average crew) (27 pts)
 (f) Desaix 74 gun Ship of the Line (average crew) (24 pts)

Lake Champlain:

Wind from the N, blowing a fresh breeze.

- (a) Saratoga 26 gun Sloop (crack crew) (12 pts)
 (a) Eagle 20 gun Sloop (crack crew) (11 pts)

- (a) Ticonderoga 17 gun Sloop (crack crew) (9 pts)
- (a) Preble 7 gun Brig (crack crew) (4 pts)
- (b) Constance 37 gun Frigate (crack crew) (14 pts)
- (b) Linnet 16 gun Sloop (elite crew) (10 pts)
- (b) Chubb 11 gun Brig (crack crew) (5 pts)

Last Voyage of the USS President:

Wind from the N, blowing a fresh breeze.

- (a) President 44 gun Frigate (elite crew) (24 pts)
- (b) Endymion 40 gun Frigate (crack crew) (17 pts)
- (b) Pomone 44 gun Frigate (crack crew) (20 pts)
- (b) Tenedos 38 gun Frigate (crack crew) (15 pts)

Hornblower and the Natividad:

Wind from the E, blowing a gale.

A scenario for you Horny fans. Remember, he sank the Natividad against heavy odds and winds. Hint: don't try to board the Natividad, her crew is much bigger, albeit green.

- (b) Lydia 36 gun Frigate (elite crew) (13 pts)
- (s) Natividad 50 gun Ship of the Line (green crew) (14 pts)

Curse of the Flying Dutchman:

Wind from the S, blowing a fresh breeze.

Just for fun, take the Piece of cake.

- (s) Piece of Cake 24 gun Corvette (average crew) (9 pts)
- (f) Flying Dutchy 120 gun 3 Decker SOL (elite crew) (43 pts)

The South Pacific:

Wind from the S, blowing a strong breeze.

- (a) USS Scurvy 136 gun 3 Decker SOL (mutinous crew) (27 pts)
- (b) HMS Tahiti 120 gun 3 Decker SOL (elite crew) (43 pts)
- (s) Australian 32 gun Frigate (average crew) (9 pts)
- (f) Bikini Atoll 7 gun Brig (crack crew) (4 pts)

Hornblower and the battle of Rosas

Wind from the E, blowing a fresh breeze.

The only battle Hornblower ever lost. He was able to dismast one ship and stern rake the others though. See if you can do as well.

- (b) Sutherland 74 gun Ship of the Line (crack crew) (26 pts)
- (f) Turenne 80 gun 3 Decker SOL (average crew) (27 pts)
- (f) Nightmare 74 gun Ship of the Line (average crew) (24 pts)
- (f) Paris 112 gun 3 Decker SOL (green crew) (27 pts)

- (f) Napoleon 74 gun Ship of the Line (green crew) (20 pts)

Cape Horn:

Wind from the NE, blowing a strong breeze.

- (a) Concord 80 gun Ship of the Line (average crew) (27 pts)
 (a) Berkeley 98 gun 3 Decker SOL (crack crew) (28 pts)
 (b) Thames 120 gun 3 Decker SOL (elite crew) (43 pts)
 (s) Madrid 112 gun 3 Decker SOL (green crew) (27 pts)
 (f) Musket 80 gun 3 Decker SOL (average crew) (27 pts)

New Orleans:

Wind from the SE, blowing a fresh breeze.

Watch that little Cypress go!

- (a) Alligator 120 gun 3 Decker SOL (elite crew) (43 pts)
 (b) Firefly 74 gun Ship of the Line (crack crew) (27 pts)
 (b) Cypress 44 gun Frigate (elite crew) (14 pts)

Botany Bay:

Wind from the N, blowing a fresh breeze.

- (b) Shark 64 gun Ship of the Line (average crew) (18 pts)
 (f) Coral Snake 44 gun Corvette (elite crew) (24 pts)
 (f) Sea Lion 44 gun Frigate (elite crew) (24 pts)

Voyage to the Bottom of the

Wind from the NW, blowing a fresh breeze.

This one is dedicated to Richard Basehart and David Hedison.

- (a) Seaview 120 gun 3 Decker SOL (elite crew) (43 pts)
 (a) Flying Sub 40 gun Frigate (crack crew) (17 pts)
 (b) Mermaid 136 gun 3 Decker SOL (mutinous crew) (27 pts)
 (s) Giant Squid 112 gun 3 Decker SOL (green crew) (27 pts)

Frigate Action:

Wind from the E, blowing a fresh breeze.

- (a) Killdeer 40 gun Frigate (average crew) (15 pts)
 (b) Sandpiper 40 gun Frigate (average crew) (15 pts)
 (s) Curlew 38 gun Frigate (crack crew) (16 pts)

The Battle of Midway:

Wind from the E, blowing a moderate breeze.

- (a) Enterprise 80 gun Ship of the Line (crack crew) (31 pts)
 (a) Yorktown 80 gun Ship of the Line (average crew) (27 pts)
 (a) Hornet 74 gun Ship of the Line (average crew) (24 pts)

- (j) Akagi 112 gun 3 Decker SOL (green crew) (27 pts)
- (j) Kaga 96 gun 3 Decker SOL (green crew) (24 pts)
- (j) Soryu 80 gun Ship of the Line (green crew) (23 pts)

Star Trek:

Wind from the S, blowing a fresh breeze.

- (a) Enterprise 450 gun Ship of the Line (elite crew) (75 pts)
- (a) Yorktown 450 gun Ship of the Line (elite crew) (75 pts)
- (a) Reliant 450 gun Ship of the Line (elite crew) (75 pts)
- (a) Galileo 450 gun Ship of the Line (elite crew) (75 pts)
- (k) Kobayashi Maru 450 gun Ship of the Line (elite crew) (75 pts)
- (k) Klingon II 450 gun Ship of the Line (elite crew) (75 pts)
- (o) Red Orion 450 gun Ship of the Line (elite crew) (75 pts)
- (o) Blue Orion 450 gun Ship of the Line (elite crew) (75 pts)

IMPLEMENTATION

sail is really two programs in one. Each player starts up a process which runs his own ship. In addition, a driver process is forked (by the first player) to run the computer ships and take care of global bookkeeping.

Because the driver must calculate moves for each ship it controls, the more ships the computer is playing, the slower the game will appear.

If a player joins a game in progress, he will synchronize with the other players (a rather slow process for everyone), and then he may play along with the rest.

To implement a multi-user game, the communicating processes must use a common temporary file as a place to read and write messages. In addition, a locking mechanism must be provided to ensure exclusive access to the shared file. For example, **sail** uses a temporary file named `/tmp/#sailsink.21` for scenario 21, and corresponding file names for the other scenarios. To provide exclusive access to the temporary file, **sail** uses a technique stolen from an old game called "pubcaves" by Jeff Cohen. Processes do a busy wait in the loop

```
for (n = 0; link(sync_file, sync_lock) < 0 && n < 30; n++)
    sleep(2);
```

until they are able to create a link to a file named `/tmp/#saillock.??`. The "??" correspond to the scenario number of the game. Since UNIX guarantees that a link will point to only one file, the process that succeeds in linking will have exclusive access to the temporary file.

CONSEQUENCES OF SEPARATE PLAYER AND DRIVER

When players do something of global interest, such as moving or firing, the driver must coordinate the action with the other ships in the game. For example, if a player wants to move in a certain direction, he writes a message into the temporary file requesting the driver to move his ship. Each "turn," the driver reads all the

messages sent from the players and decides what happened. It then writes back into the temporary file new values of variables, etc.

The most noticeable effect this communication has on the game is the delay in moving. Suppose a player types a move for his ship and hits return. What happens then? The player process saves up messages to be written to the temporary file in a buffer. Every 7 seconds or so, the player process gets exclusive access to the temporary file and writes out its buffer to the file. The driver, running asynchronously, must read in the movement command, process it, and write out the results. This takes two exclusive accesses to the temporary file. Finally, when the player process gets around to doing another 7 second update, the results of the move are displayed on the screen. Hence, every movement requires four exclusive accesses to the temporary file (anywhere from 7 to 21 seconds depending upon asynchrony) before the player sees the results of his moves.

In practice, the delays are not as annoying as they would appear. There is room for "pipelining" in the movement. After the player writes out a first movement message, a second movement command can then be issued. The first message will be in the temporary file waiting for the driver, and the second will be in the file buffer waiting to be written to the file. Thus, by always typing moves a turn ahead of the time, the player can sail around quite quickly.

If the player types several movement commands between two 7 second updates, only the last movement command typed will be seen by the driver. Movement commands within the same update "overwrite" each other, in a sense.

NAME

scramble – turn your screen into a scramble puzzle

SYNOPSIS

scramble [**-den**] [**-c num**] [**-i num**] [**-p sec**] [**-x size**] [**-y size**]
[*message-line ...*]

DESCRIPTION

scramble turns your bitmap screen into a scrambling puzzle. If you solve the puzzle, you reclaim control of the screen. You can also quit by typing <EXIT> or CTRL/Q.

If you specify any *message-line* arguments, **scramble** displays the text in the empty square, one line per *message-line*. To include spaces in a single *message-line*, use backslash (\) or double-quote (") in the conventional manner.

Note: It is not nice to **crp**(1) or **rlogin**(1) onto other people's nodes and run this, since it scrambles their screens while they're trying to do important work (wink, wink).

OPTIONS

- d** Run in the current DM window, not on the entire screen.
- e** If the scrambling is to stop, just exit when done; don't give the player a chance to solve it.
- n** Don't number the pieces.
- c num** Scramble *num* times, then stop.
- i num** Keep scrambling the puzzle until you hit a recognized key (see below), but scramble a minimum of *num* times (default 0).
- p sec** After you solve the puzzle, pause for *sec* seconds before exiting so you can admire your skill (default 0).
- x size** Make the puzzle *size* squares across (default 5).
- y size** Make the puzzle *size* squares high (default 4).

COMMANDS

- <EXIT>** Quit **scramble**.
- ←, →, ↑, ↓** Move a square into the empty box from the right, the left, above, or below, respectively.
- ⌞←, →⌟** Move the entire row into the empty box from the right or left, respectively.
- Boxed arrow keys Move the empty box to the left, to the right, up, or down.
- Mouse buttons Clicking any mouse button with the cursor in a square shifts the entire row from that piece onward towards the empty square. If the cursor is in neither the row nor the column of the empty square, this has no effect.

NAME

snake, snscore – display chase game

SYNOPSIS

```
/usr/games/snake [ -wn ] [ -ln ]
/usr/games/snscore
```

DESCRIPTION

snake is a display-based game which must be played on a **termcap(5)** supported screen. The object of the game is to make as much money as possible without getting eaten by the snake. The **-l** and **-w** options allow you to specify the length and width of the field. By default the entire screen (except for the last column) is used.

You are represented on the screen by an **I**. The snake is 6 squares long and is represented by **S**'s. The money is **\$**, and an exit is **#**. Your score is posted in the upper left hand corner.

You can move around using the same conventions as **vi(1)**, the **h**, **j**, **k**, and **l** keys work, as do the arrow keys. Other possibilities include:

sefc These keys are like **hjkl** but form a directed pad around the **d** key.

HJKL These keys move you all the way in the indicated direction to the same row or column as the money. This does *not* let you jump away from the snake, but rather saves you from having to type a key repeatedly. The snake still gets all his turns.

SEFC Likewise for the upper case versions on the left.

ATPB These keys move you to the four edges of the screen. Their position on the keyboard is the mnemonic, e.g. **P** is at the far right of the keyboard.

x This lets you quit the game at any time.

p Points in a direction you might want to go.

w Space warp to get out of tight squeezes, at a price.

! Shell escape

CTRL/Z

Suspend the snake game, on systems which support it. Otherwise an interactive shell is started up.

To earn money, move to the same square the money is on. A new **\$** will appear when you earn the current one. As you get richer, the snake gets hungrier. To leave the game, move to the exit (**#**).

A record is kept of the personal best score of each player. Scores are only counted if you leave at the exit, getting eaten by the snake is worth nothing.

As in pinball, matching the last digit of your score to the number which appears after the game is worth a bonus.

To see who wastes time playing snake, run **/usr/games/snscore**.

FILES

/usr/games/lib/snakerawscores

database of personal bests

`/usr/games/lib/snake.log`
`/usr/games/busy`

log of games played
program to determine if system too busy

BUGS

When playing on a small screen, it's hard to tell when you hit the edge of the screen. The scoring function takes into account the size of the screen. A perfect function to do this equitably has not been devised.

NAME

strfile, **unstr** – create a random access file for storing strings

SYNOPSIS

strfile [*options*] *sourcefile* [*datafile*]

unstr [*-o*] [*-cx*] *datafile* [*outfile*]

DESCRIPTION

strfile converts a file containing a set of strings into a data file which contains those strings along with a seek pointer table to the beginning of each. This allows random access to the strings. **strfile**'s main use is to add entries to the **fortune(6)** database.

strfile creates *datafile* from a *sourcefile* that consists of strings separated by lines starting with “%%” or “%-”. Anything following these characters on the line will be ignored, so comments can be placed on these lines. A “%%” simply separates strings; a “%-” separates not only strings but sections. A file can have up to four sections (in other words, up to three delimiters). This can be used in a program-defined way.

If you do not specify a *datafile* on the command line, **strfile** creates a file named *sourcefile.dat*. The *datafile* contains a header describing its contents, the seek pointers to the beginning of each string, and the strings themselves (terminated by null bytes).

The format of the header is

```
# define  MAXDELIMS3

# define  STR_RANDOM                0x1
# define  STR_ORDERED              0x2

typedef struct {
    unsigned long  str_numstr;        /* # of strings in the file */
    unsigned long  str_longlen;      /* length of longest string */
    unsigned long  str_shortlen;     /* length of shortest string */
    long           str_delims[MAXDELIMS]; /* delimiter markings */
    off_t          str_dpos[MAXDELIMS]; /* delimiter positions */
    short          str_flags;        /* bit field for flags */
} STRFILE;
```

The values in **str_delims** are the indices of the first string which follows each “%-” in the file. The field **str_flags** has the bit **str_random** set if the **-r** flag was specified, or **str_ordered** if the **-o** flag was specified.

unstr undoes the work of **strfile**. It serves primarily as an emergency backup in case you accidentally delete your source file but still have your data file. **unstr** reads a data file and creates a corresponding output file of raw strings and delimiters.

You can invoke **unstr** with the name of the data file, the name of the output file, or both. If you specify both, **unstr** treats them literally as the input and output files. If you provide a single argument ending in “.dat,” **unstr** assumes this to be the data

file, and writes its output to that filename stripped of the `“.dat”` suffix. If the single argument doesn't end in `“.dat,”` the program treats this as the name of the output file, and consequently reads its input from *outputfile.dat*.

If you want a character other than `“%”` as your delimiter, use the `-c` option to change it.

unstr normally prints out the strings in the order they occur in the data file. If you give it the `-o` option, it will write them out in the seek pointer order, which is different if the file was randomized or alphabetized when created. Using this option, you can create sorted versions of your input file by using **strfile -o**, and then using **unstr -o** to dump them out in the table order.

OPTIONS

Following are the options for **strfile**. Only the `-o` and `-c` options may be used with **unstr**.

- `-` Print a usage summary.
- `-cx` Use character *x* as the delimiter instead of `%`.
- `-s` Run silently; do not summarize processing at the end of the run.
- `-v` Use verbose mode; summarize processing at the end of the run (default).
- `-o` Order the strings alphabetically. **strfile** stores the strings in the same order in the data file as in the source, but the seek pointer table will be sorted in alphabetical order of the strings pointed to. Any initial non-alphanumeric characters are ignored. This sets the `str_ordered` bit in the `str_flags` field of the header.
- `-i` Ignore case when ordering.
- `-r` Randomize the order of the seek pointers in the table. The strings will be stored in the same order in *datafile* as in *sourcefile*, but the seek pointer table will be randomized. This sets the `str_random` bit in the `str_flags` field of the header.

EXAMPLES

To convert a file called **scene** which consists of lines like

```
%%
Hofstadter's Law:
    It always takes longer than you expect, even when you take
    Hofstadter's Law into account.
%%
"It is bad luck to be superstitious."
    -- Andrew W. Mathis
%%
If A = B and B = C, then A = C, except where void or prohibited by law.
    -- Roy Santoro
```

use the following command:

```
% strfile scene
"scene" converted to "scene.dat"
There were 1168 strings
Longest string: 1156 bytes
Shortest string: 0 bytes
```

FILES

strfile.h header file

SEE ALSO

fortune(6)

NAME

teachgammon – teach the game of backgammon

SYNOPSIS

/usr/games/teachgammon

DESCRIPTION

This program teaches you the rules for backgammon, supplies hints on strategy, and provides a tutorial consisting of a practice game against the computer.

FILES

/etc/termcap terminal capability database

SEE ALSO

backgammon(6)

NAME

trek – trekkie game

SYNOPSIS

/usr/games/trek [[**-a**] *file*]

DESCRIPTION

trek is a game of space glory and war.

If a filename is given, a log of the game is written into that file. If the **-a** flag is given before the filename, that file is appended to, not truncated.

trek asks what length game you would like. Valid responses are “short”, “medium”, and “long”. You may also type “restart”, which restarts a previously saved game. The program then prompts for the skill level, to which you must respond “novice”, “fair”, “good”, “expert”, “commodore”, or “impossible”. You should normally start out with “novice” and work up.

In general throughout the game, if you forget what is appropriate the game will tell you what it expects if you type a question mark.

COMMAND SUMMARY

abandon	capture
cloak up/down	
computer request; ...	damages
destruct	dock
help	impulse course distance
lrscan	move course distance
phasers automatic amount	
phasers manual amt1 course1 spread1 ...	
torpedo course [yes] angle/no	
ram course distance	rest time
shell	shields up/down
srscan [yes/no]	
status	terminate yes/no
undock	visual course
warp warp_factor	

NAME

ttt - tic-tac-toe

SYNOPSIS

/usr/games/ttt

DESCRIPTION

ttt is the popular X and O game, but it is also a learning program that never makes the same mistake twice.

When you begin, the program prompts you with

Accumulated knowledge? (Yes or No)

You must respond with a yes or no. The program tells you how many "words" of knowledge it currently has in its learning file. Then **ttt** states whether or not this is a new game, prints a three-by-three grid of numbers 1-9, and prompts with "Your move?" as shown below:

new game

123

456

789

Your move?

As you specify numbers in the grid, the program places an "X" in the location where the number once was. After you move, the program takes a turn, placing an "O" in an available spot. This continues until either you or the program wins by creating a vertical, horizontal, or diagonal line of three of the same characters in a row on the grid. To exit the game, type an interrupt.

Although it learns, **ttt** learns slowly. It must lose nearly 80 games to completely know the game.

FILES

/usr/games/lib/ttt.a learning file

NAME

vine – grow vines

SYNOPSIS

/usr/games/vine [*options*]

DESCRIPTION

vine creates interesting growing things on your screen. To halt the kudzu-like onslaught, type an interrupt.

OPTIONS

- e** Grow vines along the edge of the window.
- c** Grow vines from the center of the window.
- h** Exit if the RETURN key is pressed.
- r** Display in reverse video.
- i[*num*]** Interleave black/white leaves, one black for every *num* white ones. If *num* is negative, perform white/black interleaving.
- s[*size*]** Set maximum size of leaves (default is 2).
- d *device*** Use alternate device. Possible values for *device* are **borrow** (use entire display), **borrow_nc** (same as **borrow** but doesn't clear screen first), **direct** (in current window only), and **bg** (use display background).
- v[*num*]** Grow *num* vines from the top of the window.
- bnum** Specify *num* as probability of branching (default 5).
- lnum** As *num* increases (to maximum 35), more leaves grow from each stem.
- Rnum** Specify *num* different leaf directions; rotate *num* degrees for each leaf.
- q** Display usage information.

EXAMPLE

An especially pretty liana can be generated with

vine -e -i2

CAUTION

The **-c** option creates immense numbers of forks, perhaps taking over your machine for lengthy periods.

NAME

worm – play the growing worm game

SYNOPSIS

/usr/games/worm [*size*]

DESCRIPTION

In **worm**, you are a little worm; your body is the string of "o"'s on the screen and your head is the "@". You move with the "hjkl" keys as in the game **snake**(6). If you don't press any keys, you continue in the direction you last moved. The uppercase "HJKL" keys move you as if you had pressed the corresponding lowercase key several times (9 for HL and 5 for JK) unless you run into a digit, in which case you stop.

On the screen you will see a digit. If your worm eats the digit, it will grow longer; the actual amount depends on the digit eaten. The object of the game is to see how long you can make the worm grow.

The game ends when the worm runs into the sides of the screen or itself. The current score (how much the worm has grown) is shown in the upper left corner of the screen.

The optional argument, if present, sets the initial length of the worm.

BUGS

If the initial length of the worm is set to less than 1 or more than 75, various strange things happen.

NAME

worms - animate worms on a display terminal

SYNOPSIS

/usr/games/worms [*options*]

DESCRIPTION

worms generates squirming annelid-like creatures on your display. This is an adaptation of an older program called **WORM**.

OPTIONS

- field** Make a "field" for the worm(s) to eat.
- trail** Make each worm leave a trail behind it.
- length *l*** Make each worm *l* characters long.
- number *n*** Create *n* worms.

FILES

/etc/termcap

BUGS

The lower right-hand character position will not be updated properly on a terminal that wraps at the right margin.

Terminal initialization is not performed.

NAME

wump – the game of hunt-the-wumpus

SYNOPSIS

/usr/games/wump

DESCRIPTION

wump plays the game of 'Hunt the Wumpus.' A Wumpus is a creature that lives in a cave with several rooms connected by tunnels. You wander among the rooms, trying to shoot the Wumpus with an arrow, meanwhile avoiding being eaten by the Wumpus and falling into Bottomless Pits. There are also Super Bats which are likely to pick you up and drop you in some random room.

The program asks various questions which you answer one per line; it will give a more detailed description if you want.

This program is based on one described in *People's Computer Company*, 2, 2 (November 1973).

Section 7
Miscellaneous

The first of these is the fact that the
 government has been unable to
 maintain a stable currency. This
 has led to a loss of confidence
 in the government and a
 consequent loss of support
 from the people. The second
 is the fact that the government
 has been unable to maintain
 a stable economy. This has
 led to a loss of confidence
 in the government and a
 consequent loss of support
 from the people. The third
 is the fact that the government
 has been unable to maintain
 a stable society. This has
 led to a loss of confidence
 in the government and a
 consequent loss of support
 from the people.

NAME

miscellaneous – miscellaneous useful information pages

DESCRIPTION

This section contains miscellaneous documentation, mostly in the area of text processing macro packages for **troff**(1).

acl	description of Domain/OS Access Control Lists (ACLs)
ascii	map of ASCII character set
environ	user environment
eqnchar	special character definitions for eqn
hier	file system hierarchy
mailaddr	mail addressing description
man	macros to typeset manual pages
me	macros for formatting papers
ms	macros for formatting manuscripts
term	conventional names for terminals

SEE ALSO

domain(7)

NAME

domain – miscellaneous Domain/OS features

DESCRIPTION

While providing all of the significant functionality of 4.3 BSD, Domain/OS BSD actually represents only a subset of the greater functionality of Domain/OS. Furthermore, Domain/OS BSD omits some features of 4.3 BSD that are irrelevant to Apollo® workstations. The following paragraphs describe miscellaneous aspects of that greater functionality that are visible.

Pages that describe Domain/OS-specific commands are identified with the heading, “Domain/OS BSD”; standard BSD command pages are identified with the heading “BSD”.

acl(7) describes Access Control Lists (ACLs), the Domain/OS file access scheme. Domain/OS BSD derives its access and protection modes from this scheme (see **intro(2)** and *Managing BSD System Software*).

environ(7) includes descriptions of additional Domain/OS environment variables.

hier(7) describes the additional directories in the Domain/OS system hierarchy.

term(7) includes Apollo terminal names.

SEE ALSO

domain(1), **domain(2)**, **intro(2)**, **domain(3)**, **domain(4)**,
Managing BSD System Software;
Using Your BSD Environment.

NAME

acl – details about access control lists (ACLs)

DESCRIPTION

Every object in the system (whether directory or file) has an access control list (ACL) that defines who may access that object, and in what ways. The ACL is made up of a series of entries that consist of two elements: a subject identifier (SID) and a set of rights. Each entry gives one subject the right to perform some operations (read, write, execute, etc) on the object that the ACL protects.

There are required ACL entries and extended ACL entries. Each object must have an ACL entry for an owner (user), group, organization, and others (world). You can also create extended ACL entries that specify permissions for other users, groups, and organizations. Required ACL entries are listed before extended entries. ACL entries are automatically arranged in order, from the most to the least specific subject identifier. For example, the ACLs for individuals appear before the ACLs for others.

Standard UNIX permissions use required entries, with no extended ACL entries.

SUBJECT IDENTIFIERS

The subject identifier (SID) identifies those users to whom the specified set of rights apply. The subject identifier is in the format:

user.group.organization

For example:

barb.none.r_d

User, group, and organization specify names that are in the associated network registry files. You may use the wildcard % in any of the "ugo" fields.

ACCESS RIGHTS

You may assign the following access rights to the types of objects indicated:

Any object:

p Protect rights; allow rights to be changed

Files:

r Read rights

w Write rights

x Execute rights

k Keep rights; prevents an object from being deleted or from having its name changed

Directories:

w Write rights; allows names to be added, changed or deleted

r Read rights; allows directory to be listed

x Execute/search rights; allows directories to be searched for subordinate objects.

k Keep rights; prevents an object from being deleted or from having its name changed

SPECIFYING ACCESS RIGHTS

You may specify access rights individually or in groups. Each object has the required entries of User, Group, Organization, and World. However, it is sometimes useful to have these specified, but not used for rights checking. This may be done by specifying the **I** (ignore) mode with the **chacl(1)** command. The **I** mode is only valid for required entries.

Objects that are part of protected subsystems indicate this when their ACLs are displayed.

NOTE: To delete a tree you need write rights to any object being deleted. If objects are protected with keep rights, you must have protect rights as well.

ACLS AND DIRECTORIES

In addition to its own ACL, each directory contains two additional ACLs (called initial ACLs): one for new files and another for new subdirectories created within that directory. When you create a new file or directory, or copy one to a new location in the file hierarchy, the system assigns an ACL to it by copying the appropriate initial ACL stored in the parent directory. When the newly created object is a directory, the two initial ACLs from the parent are replicated in the new subdirectory, unless you specifically indicate otherwise (see the **chacl(1)** command). The various options on the **chacl(1)**, **cpacl(1)**, and **lsacl(1)** commands determine which of these access control lists you are editing, copying or displaying.

Initial ACLs may be either completely specified in the directory, or may inherit information from the process which is creating the object. Normal UNIX behavior is implemented by using inheritance from the process.

Either the subject identifier information in a required entry or the rights information in a required entry may be inherited (or both). The options to the **chacl** command allow the inheritance to be specified.

EXAMPLE

The ACL entries for a file might look like the following example.

Required entries:

SID	Rights	Meaning
mary.%.%	prwx--	Owner rights
%.none.%	[Ignored]	No group listed
%.%.r_d	--r-x-	r_d has read and execute rights
%.%.%	-----	No others have rights to this file

Extended ACL entries:

SID	Rights	Meaning
jill.%.%	prwx--	Additional user's rights
%.backup.%	-r----	The backup group has read rights

SEE ALSO

chacl(1), **cpacl(1)**, **lsacl(1)**, **salacl(8)**

NAME

ascii – map of ASCII character set

SYNOPSIS

cat /usr/pub/ascii

DESCRIPTION

ascii is a map of the ASCII character set, to be printed as needed. 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

environ – user environment variables

SYNOPSIS

```
extern char **environ;
```

DESCRIPTION

The **execve(2)** call makes an array of strings called the “environment” available when a process begins. By convention, these strings have the form *name=value*.

VARIABLE NAMES

The following names are used by various commands:

- | | |
|--------------|--|
| PATH | The sequence of directory prefixes that sh(1) , time(1) , nice(1) , etc., apply in searching for a file known by an incomplete path name. The prefixes are separated by colons (:). login(1) sets PATH=/usr/ucb:/bin:/usr/bin:/usr/apollo/bin . |
| HOME | Name of the user's login directory, set by login(1) from the password file passwd(5) . |
| SHELL | The filename of the user's login shell. |
| TERM | The kind of terminal for which output is to be prepared. This information is used by commands, such as vi(1) , nroff(1) , or plot(1G) , which may exploit special terminal capabilities. See /etc/termcap (termcap(5)) for a list of terminal types. |
| TERMCAP | The string describing the terminal in TERM , or the name of the termcap file, see termcap(5) , termcap(3X) . |
| EXINIT | A startup list of commands read by ex(1) , edit(1) , and vi(1) . |
| USER | The login name of the user. |
| PRINTER | The name of the default printer to be used by lpr(1) , lpq(1) , and lprm(1) . |
| PROJECT | Project associated with login name. |
| ORGANIZATION | Organization associated with login name. |
| NODEID | Hexadecimal ID of current node, for example, 2b1f . |
| NODETYPE | Model number of the workstation on which the process is running, for example, dn3000 . |
| SYSTYPE | Version of the Domain/OS environment currently set as systype, for example, bsd4.3 . |
| TZ | Time zone information. The format is XXXNZZZ where XXX is standard local time zone abbreviation, N is the difference in hours from Greenwich Mean Time, and ZZZ is the abbreviation for the daylight-saving local time zone, if any; for example, EST5EDT . |

APOLLO_STATUS

Prints additional Apollo status information anytime **perror()** is called.

ISP Instruction set processor type. The value of this variable determines the resolution of variant links in `/usr/lib` having names of the general form

linkname → `isp_$(ISP)/filename`

The *linknames* are those of the various bindable libraries and C initialization routines used by `ld` and `f77`: `crt0.o`, `libF77.a`, `libld.a`, and so on. When `ISP=m68k`, these variant links resolve to files of the corresponding names in `/usr/lib/isp_m68k`, a directory containing bindable libraries and C initialization routines for M680X0-based workstations. When `ISP=a88k`, these variant links resolve to files of the corresponding names in `/usr/lib/isp_a88k`, a directory containing bindable libraries and C initialization routines for the Series 10000.

`ld` and `f77` set `ISP` according to the option you specify to the `-cpu` switch, allowing you to link and compile either for M680X0-based workstations (`-cpu m68k`) or for the Series 10000 (`-cpu a88k`).

Additional names may be placed in the environment by the `export` command and the `name=value` arguments in `sh(1)` and `ksh(1)`, or by the `setenv` command in `csh(1)`. Arguments may also be placed in the environment at the point of an `execve(2)`.

NOTES

It is unwise to use the names `MAIL`, `PS1`, `PS2`, `IFS`. They conflict with certain `sh(1)` variables that are frequently exported by `.profile` files.

When you log in, the Display Manager looks for a `~/environ` file, and uses it to set the environment and the `SYSTYPE` variable. If this file doesn't exist, the node defaults are used. The `/etc/environ` file establishes the default environment and `SYSTYPE` variable for the node.

SEE ALSO

`csh(1)`, `environment(1)`, `ex(1)`, `f77(1)`, `ksh(1)`, `ld(1)`, `login(1)`, `sh(1)`, `execve(2)`, `system(3)`, `termcap(3X)`, `termcap(5)`, `systype(1)`;
Managing BSD System Software;
Using Your BSD Environment.

NAME

eqnchar — special character definitions for **eqn**

SYNOPSIS

eqn /usr/pub/**eqnchar** [*files*] | **troff** [*options*]

neqn /usr/pub/**eqnchar** [*files*] | **nroff** [*options*]

DESCRIPTION

eqnchar contains **troff** and **nroff** character definitions for constructing various special characters. These definitions are primarily intended for use with **eqn** and **neqn**. **eqnchar** contains definitions for the following characters

<i>ciplus</i>	⊕			<i>square</i>	□
<i>citimes</i>	⊗	<i>langle</i>	/	<i>circle</i>	○
<i>wig</i>	~	<i>rangle</i>	/	<i>blot</i>	◻
<i>-wig</i>	≈	<i>hbar</i>	ℏ	<i>bullet</i>	•
<i>>wig</i>	≥	<i>ppd</i>	⊥	<i>prop</i>	∞
<i><wig</i>	≤	<i><-></i>	↔	<i>empty</i>	∅
<i>=wig</i>	≡	<i><=></i>	↔	<i>member</i>	∈
<i>star</i>	*	<i> <</i>	⋈	<i>nomem</i>	∉
<i>bigstar</i>	⋆	<i> ></i>	⋈	<i>cup</i>	∪
<i>=dot</i>	≐	<i>ang</i>	∠	<i>cap</i>	∩
<i>orsign</i>	∨	<i>rang</i>	⌢	<i>incl</i>	⊆
<i>andsign</i>	∧	<i>3dot</i>	⋮	<i>subset</i>	⊂
<i>=del</i>	≠	<i>thf</i>	∴	<i>supset</i>	⊃
<i>oppA</i>	∇	<i>quarter</i>	¼	<i>!subset</i>	⊄
<i>oppE</i>	≡	<i>3quarter</i>	¾	<i>!supset</i>	⊅
<i>angstrom</i>	Å	<i>degree</i>	°		

FILES

/usr/pub/**eqnchar**

SEE ALSO

troff(1), **eqn**(1)

NAME

hier – file system hierarchy

DESCRIPTION

The following outline gives a quick tour through a representative Domain/OS BSD system directory hierarchy. Note that we start at the node (rather than the network) root.

/	Root directory of the local disk
/bsd4.3/	Root of the bsd4.3 tree
/sys5.3/	System V stub (provided for solution supplier only)
/lost+found	Directory for connecting detached files for salvol (8)
/dev/	Device files; a link to /sys/node_data/dev (these files are documented in section 4 of the <i>BSD Programmer's Reference</i>).
pad*	DM pads
sio?	Serial I/O ports, see sio (4)
ptyp? , ttyp?	pty 's, see pty (4)
rmt?	Magtape descriptors
rct?	Cartridge tape descriptors
console	Main console, see tty (4)
tty*	Terminals, see tty (4)
hp*	Disks, see hp (4)
...	
/bin/	Utility programs documented in section one of the <i>BSD Command Reference</i> . (See also /usr/bin/ , /usr/ucb/bin/ , /usr/apollo/bin/ .) On most nodes, /bin is the variant link \$(systype)/bin , which resolves to either /bsd4.3/bin or /sys5.3/bin , based on the current value of the SYSTYPE environment variable.
cc	C compiler
csh	C shell
ed	Line editor
sed	Stream editor
...	
/lib/	Object libraries and other stuff, see also /usr/lib/ , /usr/apollo/lib .
/doc/	Release notes and update procedures documentation.
/etc/	System files, essential data, maintenance and administration utilities. The /etc directory contains directories, files and links (variant links through \$(systype)/ for environment-specific files, and links to /sys/node_data/etc for node-specific files).
bsd4.3/	Commands specific to the BSD environment
passwd	Password file, passwd (5)
group	Group file, group (5)
motd	Message of the day, login (1)
getty	Part of login , getty (8)
init	The parent of all processes, init (8)
...	

The following files are links to **/sys/node_data/**

fstab	File system configuration table fstab(5)
mtab	Mounted file table, mtab(5)
ttys	Properties of terminals, ttys(5)
rc	Shell program to bring the system up
rc.local	Site dependent portion of rc
rc.user	Shell script to start Domain/OS BSD servers (run by rc)

...

The following files are links to **\$(systype)/**

termcap	Description of terminal capabilities, termcap(5)
cron	The clock daemon, cron(8)
mount	mount(8)

...

/sys/ Domain/OS system programs and information.

boot System boot program

node_data/ Per-node information

etc/rc Per-node boot-time script

tmp Temporary files, link to **/tmp**

usr/tmp Temporary files, link to **/usr/tmp**

startup.* DM startup command script

...

dm/ The Display Manager and screen fonts

dm Display Manager program

fonts Screen fonts

keys Key definitions files

input DM input pad

output DM output pad

...

mbx/ Interprocess communications mailboxes

spm/ The server process manager and associated files

traits/ Trait information for streams managers

types File type information for streams managers

logfiles

...

/tmp/ Temporary files, usually a link to **/sys/node_data/tmp**

/usr/ General-purpose directory, usually a mounted file system

adm/ Administrative information

apollo/ Domain/OS extensions to the UNIX environments

bin/ Domain/OS commands for the UNIX environments

lib/ Binaries and related files

include/ Domain/OS system call C include files

bin/ Additional utility programs, to keep **/bin/** small

	tmp/	Temporaries; a link to /sys/node_data/usrtmp
	dict/	Word lists, etc.
	games/	Collection of recreational and educational diversions
	lib/	Library of stuff for games
include/		Standard #include files
lib/		Object libraries
	atrun	Scheduler for at(1)
	...	
	struct/	Passes of struct(1)
	...	
	tmac/	Macros for troff(1)
	font/	Fonts for troff(1)
	uucp/	Programs and data for uucp(1C)
man/		Reference manual entries
	cat?/	Preformatted pages for section ?
new/		Binaries of new versions of programs
spool/		Delayed execution files
mail/		Mailboxes for mail(1) , a link to spool/mail
uucp/		Work files and staging area for uucp(1C)
ucb/		Source for programs in /usr/ucb
games/		Source for /usr/games
/wd/		Initial working directory of a user, typically wd is the user's login name
	.profile	Set environment for sh(1) , environ(7)
	.login	Set initial environment for csh(1)
	.cshrc	Startup file for csh(1)
	.logout	Logout file for csh(1)
	.exrc	Startup file for ex(1)
	.mailrc	Startup file for mail(1)
	user_data/	Personal DM startup files, key definitions, etc.

BUGS

The position of files is subject to change without notice.

SEE ALSO

ls(1), **apropos(1)**, **whatis(1)**, **whereis(1)**, **finger(1)**, **which(1)**, **find(1)**, **grep(1)**;
Managing BSD System Software;
Using Your BSD Environment.

NAME

limits – Domain/OS limits

DESCRIPTION

The table below lists the maximum values (unless otherwise indicated) for a wide variety of Domain/OS features.

FEATURE	SR9.5/SR9.7	SR10.1	SR10.2
-----	-----	-----	-----
Network:			
nodes/Apollo Token Ring	1000	1000	1000
nodes/IEEE 802.3	n/a	n/a	1024
nodes/IEEE 802.5	n/a	260	260
networks/Domain internet	64	64	64
hops/Domain internet	15	15	15
IPC sockets/node	23	23	64
(May be lower depending on current conditions on the node.)			
DM:			
windows	50	50	110
pads	40	40	100
lines/pad	262143	262143	262143
chars/line	256	1024	1024
paste buffers	100	100	100
fonts/pad	100	100	100
chars/keydef	256	1024	1024
Aegis Shell:			
command arguments	128	128	128
string token line length (in chars)	1024	1024	1024
pipes	~110	~110	~110
here-documents per script	~120	~120	~120
shell recursions	~160	~160	~160
(May vary depending on process depth, script size, disk and memory limitations.)			
BSD & SysV:			
length of arguments to exec	5120	10240	10240
integer value of a "char" (maximum):	127	127	127
integer value of a "char" (minimum)	-128	-128	-128
processes/user id	24	56	56 (68k) 247 (88k)
file size (in bytes)	4294967295	4294967295	4294967295

LIMITS(7)

Domain/OS BSD

LIMITS(7)

links to one file	65535	65535	65535
chars/password	8	8	8
open files/process	128	128	128
process id value	30000	30000	30000
bytes atomic in write to a pipe	5120	5120	5120
bytes written to a pipe in a write	5120	5120	5120
bytes/physical io block (actual)	1024	1024	1024
user or group id value	30000	30000	65535
Naming Server:			
entries/directory	1300	no effective limit	no effective limit
(fewer if directory contains links)			
chars/leafname	32 (16 uc)	255 (uc or lc)	255 (uc or lc)
chars/pathname	256	1023	1023
Printing:			
pixels/plot line	16k	16k	equal to GPR main memory bit map maximum.
chars/text line	1024	1024	10000
MBX:			
channels/mailbox	128	255	255
channels/process	128	255	255
remote channels/node	128	128	128
(Configurable to be fewer by node startup file.)			
IOS:			
open streams	128	128	128
DSEE:			
	(Version 3.0)	(Version 3.2)	(Version 3.3)
simultaneous users of one library	60	(SR9.7) 60	(SR10.x) 60
simultaneous users of libraries on one node	60	60	60
branches in an element	32000	32000	32000
versions on a branch	32000	32000	32000
single version size (in lines)	200000	200000	200000
components in a model	32767	32767	32767
levels that components in a model can be nested	31	31	31
libraries referenced by one model	128	128	128

GPR:

bitmap dimensions (x,y)	8192 x 8192	8192 x 8192	8192 x 8192
bitmap planes	24	24	24
hidden memory bitmap (x,y)	224 x 224	224 x 224	224 x 224

GMR2D:

metafile size (bytes):

DN3xx, DN550	7 MB	7 MB	7 MB
DNx60	250 MB	250 MB	250 MB
DN3xxx, DN560	56 MB	56 MB	56 MB
ablocks/user	40	64	64
aclasses/user	16	64	64
vertices in polylines	1000	8190	8190
vertices in polygons	1000	8190	8190
open metafiles	16	16	16
font families	8	n/a	n/a
fonts (stroke and raster)	32	n/a	n/a
segments	65536	16#FFFFFF	16#FFFFFF

GMR3D:

metafile size (bytes):

DN3xx, DN550	7 MB	7 MB	7 MB
DNx60	250 MB	250 MB	250 MB
DN3xxx, DN560, DN570, DN580	56 MB	56 MB	56 MB
ablocks/user	40	40	40
aclasses/user	16	16	16
vertices in polylines	1024	1024	1024
vertices in polygons	1024	1024	1024
vertices in mesh	1024	1024	1024
element length (in words)	32767	32767	32767
viewports	64	64	64
path depth for picking, dynamic mode and echoing	32	32	32

GSR (future limits):

gao instruction length (in words)	32767
buffer length (in words)	16#7FFFFFFF

Dialogue:

.DPD file size (in words)	256K	2M	2M
tasks and techniques	32K	32K	32K
menu and switch entries	1368	1368	1368

/com/ed

LIMITS(7)

Domain/OS BSD

LIMITS(7)

file size (in bytes)	256,000	256,000	256,000
file length (in lines)	6,000	6,000	6,000
/com/fpat			
maximum argument length (characters)	128	512	512
/com/fpatb			
block length (in bytes)	15,000	15,000	15,000
Misc:			
user processes			
DN100x0	n/a	n/a	247
all others	24	56	56
(lower on internet routers or when NETSVC-SERVERS is used)			
mappable user segments (not protected):			
16 MB VA's	312	312	312
256 MB 460's	7423	7423	7423
256 MB 020's		230Mb	230Mb
(7360 segs.)			
64 MB 020's	1656	1656	1656
64 MB 3000's	1600	1600	1600
2 GB Turbo's	60927	60927	60927
N-key rollover on keyboard	6	6	6
stack size	256K	256K	settable (default = 256k)
entries/ACL	22	22	22

NAME

mailaddr – mail addressing description

DESCRIPTION

Mail addresses are based on the ARPANET protocol listed at the end of this manual page. These addresses are in the general format

`user@domain`

where a domain is a hierarchical dot separated list of subdomains. For example, the address

`eric@monet.berkeley.edu`

is normally interpreted from right to left: the message should go to the ARPA name tables (which do not correspond exactly to the physical ARPANET), then to the Berkeley gateway, after which it should go to the local host monet. When the message reaches monet it is delivered to the user "eric."

Unlike some other forms of addressing, this does not imply any routing. Thus, although this address is specified as an ARPA address, it might travel by an alternate route if that were more convenient or efficient. For example, at Berkeley, the associated message would probably go directly to monet over the Ethernet rather than going via the Berkeley ARPANET gateway.

Abbreviation

Under certain circumstances it may not be necessary to type the entire domain name. In general, anything following the first dot may be omitted if it is the same as the domain from which you are sending the message. For example, a user on `calder.berkeley.edu` could send to `eric@monet` without adding the `berkeley.edu` since it is the same on both sending and receiving hosts.

Certain other abbreviations may be permitted as special cases. For example, at Berkeley, ARPANET hosts may be referenced without adding the `berkeley.edu` as long as their names do not conflict with a local host name.

Compatibility

Certain old address formats are converted to the new format to provide compatibility with the previous mail system. In particular,

`user@host.ARPA`

is allowed and

`host:user`

is converted to

`user@host`

to be consistent with the `rcp(1)` command.

Also, the syntax

`host!user`

is converted to:

user@host.UUCP

This is normally converted back to the **host!user** form before being sent on for compatibility with older UUCP hosts.

The current implementation is not able to route messages automatically through the UUCP network. Until that time you must explicitly tell the mail system which hosts to send your message through to get to your final destination.

Case Distinctions

Domain names (i.e., anything after the “@” sign) may be given in any mixture of upper and lower case with the exception of UUCP hostnames. Most hosts accept any combination of case in user names, with the notable exception of MULTICS sites.

Route-addrs

Under some circumstances it may be necessary to route a message through several hosts to get it to the final destination. Normally this routing is done automatically, but sometimes it is desirable to route the message manually. Addresses which show these relays are termed “route-addrs.” These use the syntax:

<@hosta,@hostb:user@hostc>

This specifies that the message should be sent to hosta, from there to hostb, and finally to hostc. This path is forced even if there is a more efficient path to hostc.

Route-addrs occur frequently on return addresses, since these are generally augmented by the software at each host. It is generally possible to ignore all but the **user@domain** part of the address to determine the actual sender.

Postmaster

Every site is required to have a user or user alias designated “postmaster” to which problems with the mail system may be addressed.

Other Networks

Some other networks can be reached by giving the name of the network as the last component of the domain. *This is not a standard feature* and may not be supported at all sites. For example, messages to CSNET or BITNET sites can often be sent to **user@host.CSNET** or **user@host.BITNET** respectively.

BUGS

The RFC822 group syntax (**group:user1,user2,user3;**) is not supported except in the special case of **group;;** because of a conflict with old berknet-style addresses.

Route-Address syntax is grotty.

UUCP- and ARPANET-style addresses do not coexist politely.

SEE ALSO

mail(1), sendmail(8); Crocker, D. H., *Standard for the Format of Arpa Internet Text Messages*, RFC822.

NAME

man – macros to typeset manual

SYNOPSIS

nroff *-man file ...*

troff *-man file ...*

DESCRIPTION

The **man** macros are used to format UNIX manual pages.

Any text argument *t* may be zero to six words. Quotes may be used to include blanks in a “word”. If *text* is empty, special treatment is applied to the next input line containing text to be printed. In this way, **.I** may be used to italicize a whole line, or **.SM** may be followed by **.B** to make small bold text.

A prevailing indent distance is remembered between successive indented paragraphs, and is reset to the default value upon reaching a non-indented paragraph. Default units for indents *i* are ens.

Type font and size are reset to default values before each paragraph, and after processing font and size setting macros.

These strings are predefined by **-man**:

***R** ‘@’, (Reg) in **nroff**.

***S** Change to default type size.

REQUESTS

Request	Cause If no Break	If no Argument	Explanation
.B <i>t</i>	no	<i>t</i> =n.t.l.*	Text <i>t</i> is bold.
.BI <i>t</i>	no	<i>t</i> =n.t.l.	Join words of <i>t</i> alternating bold and italic.
.BR <i>t</i>	no	<i>t</i> =n.t.l.	Join words of <i>t</i> alternating bold and Roman.
.DT	no	.5i li...	Restore default tabs.
.HP <i>i</i>	yes	<i>i</i> =p.i.*	Set prevailing indent to <i>i</i> . Begin paragraph with hanging indent.
.I <i>t</i>	no	<i>t</i> =n.t.l.	Text <i>t</i> is italic.
.IB <i>t</i>	no	<i>t</i> =n.t.l.	Join words of <i>t</i> alternating italic and bold.
.IP <i>x i</i>	yes	<i>x</i> =""	Same as .TP with tag <i>x</i> .
.IR <i>t</i>	no	<i>t</i> =n.t.l.	Join words of <i>t</i> alternating italic and Roman.
.LP	yes	-	Same as .PP .
.PD <i>d</i>	no	<i>d</i> =4v	Interparagraph distance is <i>d</i> .
.PP	yes	-	Begin paragraph. Set prevailing indent to .5i.

.RE	yes	-	End of relative indent. Set prevailing indent to amount of starting .RS.
.RB <i>t</i>	no	<i>t</i> =n.t.l.	Join words of <i>t</i> alternating Roman and bold.
.RI <i>t</i>	no	<i>t</i> =n.t.l.	Join words of <i>t</i> alternating Roman and italic.
.RS <i>i</i>	yes	<i>i</i> =p.i.	Start relative indent, move left margin in distance <i>i</i> . Set prevailing indent to .5i for nested indents.
.SH <i>t</i>	yes	<i>t</i> =n.t.l.	Subhead.
.SM <i>t</i>	no	<i>t</i> =n.t.l.	Text <i>t</i> is small.
.TH <i>n c x v m</i>	yes	-	Begin page named <i>n</i> of chapter <i>c</i> ; <i>x</i> is extra commentary, for example, local , for page foot center; <i>v</i> alters page foot left, for example, 4th Berkeley Distribution ; <i>m</i> alters page head center, for example, Brand X Programmer's Manual . Set prevailing indent and tabs to .5i.
.TP <i>i</i>	yes	<i>i</i> =p.i.	Set prevailing indent to <i>i</i> . Begin indented paragraph with hanging tag given by next text line. If tag doesn't fit, place it on separate line.

* n.t.l. = next text line; p.i. = prevailing indent

FILES

/usr/lib/tmac/tmac.an
/usr/lib/tmac/tmac.an.new

BUGS

Relative indents don't nest.

SEE ALSO

troff(1), man(1)

NAME

me – macros for formatting papers

SYNOPSIS

nroff **-me** [*options*] *file* ...

troff **-me** [*options*] *file* ...

DESCRIPTION

This package of **nroff** and **troff** macro definitions provides a canned formatting facility for technical papers in various formats. When producing two-column output on a terminal, filter the output through **col(1)**.

The macro requests are defined below. Many **nroff** and **troff** requests are unsafe in conjunction with this package; however, these requests may be used with impunity after the first **.pp**:

.bp begin new page
.br break output line here
.sp *n* insert *n* spacing lines
.ls *n* (line spacing) *n*=1 single, *n*=2 double space
.na no alignment of right margin
.ce *n* center next *n* lines
.ul *n* underline next *n* lines
.sz +*n* add *n* to point size

Output of the **eqn**, **neqn**, **refer**, and **tbl(1)** preprocessors for equations and tables is acceptable as input.

REQUESTS

In the following list, “initialization” refers to the first **.pp**, **.lp**, **.ip**, **.np**, **.sh**, or **.uh** macro. This list is incomplete; see *The -me Reference Manual* for interesting details.

Request	Initial	Cause	Explanation
	Value	Break	

.(c	-	yes	Begin centered block
.(d	-	no	Begin delayed text
.(f	-	no	Begin footnote
.(l	-	yes	Begin list
.(q	-	yes	Begin major quote
.(x <i>x</i>	-	no	Begin indexed item in index <i>x</i>
.(z	-	no	Begin floating keep
.)c	-	yes	End centered block
.)d	-	yes	End delayed text
.)f	-	yes	End footnote
.)l	-	yes	End list
.)q	-	yes	End major quote
.)x	-	yes	End index item
.)z	-	yes	End floating keep
.++ <i>m H</i>	-	no	Define paper section. <i>m</i> defines the part of the paper, and can be C (chapter), A (appendix), P (preliminary; for example, abstract, table

			of contents, etc.), B (bibliography), RC (chapters renumbered from page one each chapter), or RA (appendix renumbered from page one).
.+c T	-	yes	Begin chapter (or appendix, etc., as set by .++). <i>T</i> is the chapter title.
.1c	1	yes	One column format on a new page.
.2c	1	yes	Two column format.
.EN	-	yes	Space after equation produced by eqn or neqn .
.EQ x y	-	yes	Precede equation; break out and add space. Equation number is <i>y</i> . The optional argument <i>x</i> may be I to indent equation (default), L to left-adjust the equation, or C to center the equation.
.GE	-	yes	End gremlin picture.
.GS	-	yes	Begin gremlin picture.
.PE	-	yes	End pic picture.
.PS	-	yes	Begin pic picture.
.TE	-	yes	End table.
.TH	-	yes	End heading section of table.
.TS x	-	yes	Begin table; if <i>x</i> is H table has repeated heading.
.ac A N	-	no	Set up for ACM style output. <i>A</i> is the Author's name(s), <i>N</i> is the total number of pages. Must be given before the first initialization.
.b x	no	no	Print <i>x</i> in boldface; if no argument switch to boldface.
.ba +n	0	yes	Augments the base indent by <i>n</i> . This indent is used to set the indent on regular text (like paragraphs).
.bc	no	yes	Begin new column
.bi x	no	no	Print <i>x</i> in bold italics (nofill only)
.bu	-	yes	Begin bulleted paragraph
.bx x	no	no	Print <i>x</i> in a box (nofill only).
.ef 'x'y'z' ****	no	no	Set even footer to <i>x y z</i>
.eh 'x'y'z' ****	no	no	Set even header to <i>x y z</i>
.fo 'x'y'z' ****	no	no	Set footer to <i>x y z</i>
.hx	-	no	Suppress headers and footers on next page.
.he 'x'y'z' ****	no	no	Set header to <i>x y z</i>
.hl	-	yes	Draw a horizontal line
.i x	no	no	Italicize <i>x</i> ; if <i>x</i> missing, italic text follows.
.ip x y	no	yes	Start indented paragraph, with hanging tag <i>x</i> . Indentation is <i>y</i> ens (default 5).
.lp	yes	yes	Start left-blocked paragraph.
.lo	-	no	Read in a file of local macros of the form .*x . Must be given before initialization.
.np	1	yes	Start numbered paragraph.
.of 'x'y'z' ****	no	no	Set odd footer to <i>x y z</i>
.oh 'x'y'z' ****	no	no	Set odd header to <i>x y z</i>
.pd	-	yes	Print delayed text.
.pp	no	yes	Begin paragraph. First line indented.
.r	yes	no	Roman text follows.
.re	-	no	Reset tabs to default values.
.sc	no	no	Read in a file of special characters and diacritical marks. Must be given before initialization.

.sh <i>n x</i>	-	yes	Section head follows, font automatically bold. <i>n</i> is level of section, <i>x</i> is title of section.
.sk	no	no	Leave the next page blank. Only one page is remembered ahead.
.sm <i>x</i>	-	no	Set <i>x</i> in a smaller pointsize.
.sz <i>+n</i>	10p	no	Augment the point size by <i>n</i> points.
.th	no	no	Produce the paper in thesis format. Must be given before initialization.
.tp	no	yes	Begin title page.
.u <i>x</i>	-	no	Underline argument (even in troff). (Nofill only).
.uh	-	yes	Like .sh but unnumbered.
.xp <i>x</i>	-	no	Print index <i>x</i> .

FILES

/usr/lib/tmac/tmac.e
 /usr/lib/me/*

SEE ALSO

eqn(1), troff(1), refer(1), tbl(1);
-me Reference Manual, Eric P. Allman

NAME

ms – text formatting macros

SYNOPSIS

nroff –ms [*options*] *file* ...
troff –ms [*options*] *file* ...

DESCRIPTION

This package of **nroff** and **troff** macro definitions provides a formatting facility for various styles of articles, theses, and books. When producing two-column output on a terminal or line printer, or when reverse line motions are needed, filter the output through **col**(1). All external –ms macros are defined below. Many **nroff** and **troff** requests are unsafe in conjunction with this package. However, the first four requests below may be used with impunity after initialization, and the last two may be used even before initialization:

.bp begin new page
.br break output line
.sp *n* insert *n* spacing lines
.ce *n* center next *n* lines
.ls *n* line spacing: *n*=1 single, *n*=2 double space
.na no alignment of right margin

Font and point size changes with **\f** and **\s** are also allowed; for example, **\f2word\f1** will italicize *word*. Output of the **tbl**, **eqn**, and **refer**(1) preprocessors for equations, tables, and references is acceptable as input.

REQUESTS

Macro Name	Initial Value	Break? Reset?	Explanation
.AB <i>x</i>	–	y	Begin abstract; if <i>x</i> =no don't label abstract
.AE	–	y	End abstract
.AI	–	y	Author's institution
.AM	–	n	Better accent mark definitions
.AU	–	y	Author's name
.B <i>x</i>	–	n	Make <i>x</i> bold; if no <i>x</i> , switch to boldface
.B1	–	y	Begin text to be enclosed in a box
.B2	–	y	End boxed text and print it
.BT	date	n	Bottom title, printed at foot of page
.BX <i>x</i>	–	n	Print word <i>x</i> in a box
.CM	if t	n	Cut mark between pages
.CT	–	y,y	Chapter title: page number moved to CF (TM only)
.DA <i>x</i>	if n	n	Force date <i>x</i> at bottom of page; today if no <i>x</i>
.DE	–	y	End display (unfilled text) of any kind
.DS <i>x y</i>	I	y	Begin display with keep; <i>x</i> =I, L, C, B; <i>y</i> =indent
.ID <i>y</i>	8n,.5i	y	Indented display with no keep; <i>y</i> =indent
.LD	–	y	Left display with no keep
.CD	–	y	Centered display with no keep
.BD	–	y	Block display; center entire block
.EF <i>x</i>	–	n	Even page footer <i>x</i> (3 part as for .tl)

.EH x	-	n	Even page header x (3 part as for .tl)
.EN	-	y	End displayed equation produced by eqn
.EQ x y	-	y	Break out equation; x=L, I, C; y=equation number
.FE	-	n	End footnote to be placed at bottom of page
.FP	-	n	Numbered footnote paragraph; may be redefined
.FS x	-	n	Start footnote; x is optional footnote label
.HD	undef	n	Optional page header below header margin
.I x	-	n	Italicize x; if no x, switch to italics
.IP x y	-	y,y	Indented paragraph, with hanging tag x; y=indent
.IX x y	-	y	Index words x y and so on (up to five levels)
.KE	-	n	End keep of any kind
.KF	-	n	Begin floating keep; text fills remainder of page
.KS	-	y	Begin keep; unit kept together on a single page
.LG	-	n	Larger; increase point size by two
.LP	-	y,y	Left (block) paragraph.
.MC x	-	y,y	Multiple columns; x=column width
.ND x	if t	n	No date in page footer; x is date on cover
.NH x y	-	y,y	Numbered header; x=level, x=0 resets, x=S sets to y
.NL	10p	n	Set point size back to normal
.OF x	-	n	Odd page footer x (3 part as for .tl)
.OH x	-	n	Odd page header x (3 part as for .tl)
.PI	if TM	n	Print header on 1st page
.PP	-	y,y	Paragraph with first line indented
.PT	- -	n	Page title, printed at head of page
.PX x	-	y	Print index (table of contents); x=no suppresses title
.QP	-	y,y	Quote paragraph (indented and shorter)
.R	on	n	Return to Roman font
.RE	5n	y,y	Retreat: end level of relative indentation
.RP x	-	n	Released paper format; x=no stops title on first page
.RS	5n	y,y	Right shift: start level of relative indentation
.SH	-	y,y	Section header, in boldface
.SM	-	n	Smaller; decrease point size by 2
.TA	8n,5n	n	Set tabs to 8n 16n ... (nroff) 5n 10n ... (troff)
.TC x	-	y	Print table of contents at end; x=no suppresses title
.TE	-	y	End of table processed by tbl
.TH	-	y	End multi-page header of table
.TL	-	y	Title in boldface and two points larger
.TM	off	n	UC Berkeley thesis mode
.TS x	-	y,y	Begin table; if x=H table has multi-page header
.UL x	-	n	Underline x, even in troff
.UX x	-	n	UNIX; trademark message first time; x appended
.XA x y	-	y	Another index entry; x=page or no for none; y=indent
.XE	-	y	End index entry (or series of .IX entries)
.XP	-	y,y	Paragraph with first line exdented, others indented
.XS x y	-	y	Begin index entry; x=page or no for none; y=indent
.1C	on	y,y	One column format, on a new page
.2C	-	y,y	Begin two column format

.]-	-	n	Beginning of refer reference
.]0	-	n	End of unclassifiable type of reference
.]N	-	n	N= 1:journal-article, 2:book, 3:book-article, 4:report

REGISTERS

Formatting distances can be controlled in **-ms** by means of built-in number registers. For example, this sets the line length to 6.5 inches:

```
.nr LL 6.5i
```

The following table shows number registers and their default values:

Name	Register Controls	Takes Effect	Default
PS	point size	paragraph	10
VS	vertical spacing	paragraph	12
LL	line length	paragraph	6i
LT	title length	next page	same as LL
FL	footnote length	next .FS	5.5i
PD	paragraph distance	paragraph	1v (if n), .3v (if t)
DD	display distance	displays	1v (if n), .5v (if t)
PI	paragraph indent	paragraph	5n
QI	quote indent	next .QP	5n
FI	footnote indent	next .FS	2n
PO	page offset	next page	0 (if n), ~1i (if t)
HM	header margin	next page	1i
FM	footer margin	next page	1i
FF	footnote format	next .FS	0 (1, 2, 3 available)

When resetting these values, make sure to specify the appropriate units. Setting the line length to 7, for example, will result in output with one character per line. Setting **FF** to 1 suppresses footnote superscripting; setting it to 2 also suppresses indentation of the first line; and setting it to 3 produces an **.IP**-like footnote paragraph.

The following list shows the string registers available in **-ms**; they may be used anywhere in the text:

Name	String's Function
*Q	quote (" in nroff , "" in troff)
*U	unquote (" in nroff , '' in troff)
*-	dash (-- in nroff , — in troff)
*(MO	month (month of the year)
*(DY	day (current date)
**	automatically numbered footnote
*'	acute accent (before letter)
*`	grave accent (before letter)
*^	circumflex (before letter)
*,	cedilla (before letter)
*:	umlaut (before letter)
*~	tilde (before letter)

When using the extended accent mark definitions available with **.AM**, these strings should come after, rather than before, the letter to be accented.

BUGS

Floating keeps and regular keeps are diverted to the same space, so they cannot be mixed together with predictable results.

FILES

`/usr/lib/tmac/tmac.x`
`/usr/lib/ms/x.???`

SEE ALSO

`eqn(1)`, `refer(1)`, `tbl(1)`, `troff(1)`;
UNIX Text Processing.

NAME

term – conventional names for terminals

DESCRIPTION

Certain commands use these terminal names. They are maintained as part of the shell environment (see **sh(1)**, **environ(7)**).

apollo	Any Apollo display
apollo_1280_bw	Apollo 1280-line black and white display
apollo_1280_color	Apollo 1280 line color display
apollo_800_color	Apollo 800 line color display
apollo_800_color1	Apollo 15-inch color display
apollo_15_color	Apollo 15-inch color display
apollo_color	Apollo color display
apollo_19L	Apollo 19-inch display
apollo_19I	Apollo 19-inch display
apollo_15P	Apollo 15-inch display
apollo_15p	Apollo 15-inch display
adm3a	Lear Seigler Adm-3a
2621	Hewlett-Packard HP262? series terminals
hp	Hewlett-Packard HP264? series terminals
c100	Human Designed Systems Concept 100
h19	Heathkit H19
mime	Microterm mime in enhanced ACT IV mode
1620	DIABLO 1620 (and others using HyType II)
300	DASI/DTC/GSI 300 (and others using HyType I)
33	TELETYPE Model 33
37	TELETYPE Model 37
43	TELETYPE Model 43
735	Texas Instruments TI735 (and TI725)
745	Texas Instruments TI745
dumb	Terminals with no special features
dialup	A terminal on a phone line with no known characteristics
network	A terminal on a network connection with no known characteristics
4014	Tektronix 4014
vt52	Digital Equipment Corp. VT52
vt100	Digital Equipment Corp. VT100

Consult **/etc/termcap** (see **termcap(5)**) for an up-to-date and locally correct list.

Commands whose behavior may depend on the terminal either check the value of the environment variable **TERM**, or accept arguments of the form **-Tterm**, where **term** is one of the names given above.

BUGS

The programs that ought to adhere to this nomenclature do so only fitfully.

SEE ALSO

stty(1), **tabs(1)**, **plot(1G)**, **csh(1)**, **sh(1)**, **environ(7)** **ex(1)**, **clear(1)**, **more(1)**, **ul(1)**

the 1990s, the number of people in the UK who are employed in the public sector has increased by 1.5 million, from 2.5 million in 1980 to 4 million in 1995. The public sector has become a major employer in the UK, and its growth has been a major factor in the overall growth of the economy.

The public sector has also become a major employer of women. In 1980, women made up 40% of the public sector workforce, and by 1995, this figure had risen to 50%. This increase in the number of women in the public sector has been a major factor in the overall increase in the number of women in the workforce. The public sector has also become a major employer of young people. In 1980, young people made up 10% of the public sector workforce, and by 1995, this figure had risen to 20%.

The public sector has also become a major employer of people with disabilities. In 1980, people with disabilities made up 5% of the public sector workforce, and by 1995, this figure had risen to 10%. This increase in the number of people with disabilities in the public sector has been a major factor in the overall increase in the number of people with disabilities in the workforce. The public sector has also become a major employer of people from ethnic minorities. In 1980, people from ethnic minorities made up 5% of the public sector workforce, and by 1995, this figure had risen to 10%.

The public sector has also become a major employer of people who are over 50 years old. In 1980, people over 50 years old made up 10% of the public sector workforce, and by 1995, this figure had risen to 20%. This increase in the number of people over 50 years old in the public sector has been a major factor in the overall increase in the number of people over 50 years old in the workforce. The public sector has also become a major employer of people who are under 20 years old. In 1980, people under 20 years old made up 5% of the public sector workforce, and by 1995, this figure had risen to 10%.

The public sector has also become a major employer of people who are over 65 years old. In 1980, people over 65 years old made up 5% of the public sector workforce, and by 1995, this figure had risen to 10%. This increase in the number of people over 65 years old in the public sector has been a major factor in the overall increase in the number of people over 65 years old in the workforce. The public sector has also become a major employer of people who are under 16 years old. In 1980, people under 16 years old made up 5% of the public sector workforce, and by 1995, this figure had risen to 10%.

The public sector has also become a major employer of people who are over 75 years old. In 1980, people over 75 years old made up 5% of the public sector workforce, and by 1995, this figure had risen to 10%. This increase in the number of people over 75 years old in the public sector has been a major factor in the overall increase in the number of people over 75 years old in the workforce. The public sector has also become a major employer of people who are under 12 years old. In 1980, people under 12 years old made up 5% of the public sector workforce, and by 1995, this figure had risen to 10%.

The public sector has also become a major employer of people who are over 85 years old. In 1980, people over 85 years old made up 5% of the public sector workforce, and by 1995, this figure had risen to 10%. This increase in the number of people over 85 years old in the public sector has been a major factor in the overall increase in the number of people over 85 years old in the workforce. The public sector has also become a major employer of people who are under 8 years old. In 1980, people under 8 years old made up 5% of the public sector workforce, and by 1995, this figure had risen to 10%.

The public sector has also become a major employer of people who are over 90 years old. In 1980, people over 90 years old made up 5% of the public sector workforce, and by 1995, this figure had risen to 10%. This increase in the number of people over 90 years old in the public sector has been a major factor in the overall increase in the number of people over 90 years old in the workforce. The public sector has also become a major employer of people who are under 5 years old. In 1980, people under 5 years old made up 5% of the public sector workforce, and by 1995, this figure had risen to 10%.

Reader's Response

Please take a few minutes to send us the information we need to revise and improve our manuals from your point of view.

Document Title: *BSD Command Reference*
Order No. 005800-A01

Your Name _____ Date _____

Organization _____

Street Address _____

City _____ State _____ Zip _____

Telephone number (____) _____

When you use the Apollo system, what job(s) do you perform?

- | | |
|---|--|
| ☐ Programming | ☐ Application End User |
| ☐ Hardware Engineering | ☐ System Administration |
| ☐ Other (describe) _____ | |

How many years of **experience** do you have in using the Apollo system:

What programming **languages** do you use with the Apollo system?

How would you **evaluate** this book?

	Excellent		Average		Poor
Completeness	1	2	3	4	5
Accuracy	1	2	3	4	5
Usability	1	2	3	4	5
Additional Comments: _____					

No postage necessary if mailed in the U.S.

cut or fold along dotted line

fold



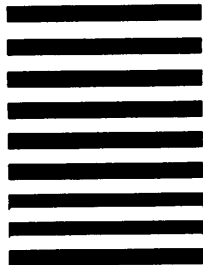
BUSINESS REPLY MAIL

FIRST CLASS MAIL PERMIT NO. 78 CHELMSFORD, MA 01824

POSTAGE WILL BE PAID BY ADDRESSEE

APOLLO COMPUTER INC.
Technical Publications
P.O. Box 451
Chelmsford, MA 01824

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES



fold

Reader's Response

Please take a few minutes to send us the information we need to revise and improve our manuals from your point of view.

Document Title: *BSD Command Reference*

Order No. 005800-A01

Your Name Date

Organization

Street Address

City State Zip

Telephone number (____) _____

When you use the Apollo system, what job(s) do you perform?

- | | |
|---|--|
| ☐ Programming | ☐ Application End User |
| ☐ Hardware Engineering | ☐ System Administration |
| ☐ Other (describe) _____ | |

How many years of experience do you have in using the Apollo system:

What programming languages do you use with the Apollo system?

How would you evaluate this book?

	Excellent		Average		Poor
Completeness	1	2	3	4	5
Accuracy	1	2	3	4	5
Usability	1	2	3	4	5
Additional Comments: _____					

No postage necessary if mailed in the U.S.

fold



NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS MAIL PERMIT NO. 78 CHELMSFORD, MA 01824

POSTAGE WILL BE PAID BY ADDRESSEE

APOLLO COMPUTER INC.
Technical Publications
P.O. Box 451
Chelmsford, MA 01824



fold