

# UniPlus+<sup>®</sup> System V, Release 2

---

Volume 3

Administrator Manual



## **COPYRIGHT**

Copyright © 1985 by UniSoft Systems. Portions of this material have been previously copyrighted by AT&T Bell Laboratories, Western Electric Company, and Regents of the University of California. Holders of a UNIX and UniPlus<sup>+</sup> software license are permitted to copy this document, or any portion of it, as necessary for licensed use of the software, provided this copyright notice and statement of permission are included.

## **DISCLAIMER**

While UniSoft Systems has endeavored to exercise care in the preparation of this guide, it nevertheless makes no warranties of any kind with regard to the documentation contained herein, including no warranty of merchantability or fitness for a particular purpose. In no event shall UniSoft be liable for incidental or consequential damages in connection with or arising out of the furnishing, performance, or use of any of this documentation.

## **TRADEMARKS**

UNIX is a trademark of AT&T Bell Laboratories. UniPlus<sup>+</sup> and UniSoft are registered trademarks of UniSoft Systems.

Adapted to UniPlus<sup>+</sup> by Heather Allen of UniSoft Systems.

# INTRODUCTION

This manual supplements the information contained in the *UniPlus<sup>+</sup> User Manual* and provides an easy reference volume for those who administer the UniPlus<sup>+</sup> operating system. Only those commands and descriptions appropriate for the system administrator are included.

This manual is divided into three sections:

1. System Maintenance Commands and Application Programs
7. Special Files
8. System Maintenance Procedures

Throughout this volume, each reference of the form *name(1M)*, *name(7)*, or *name(8)*, refers to entries in this manual, while all other references to entries of the form *name(N)*, where N is a number or a number followed by a letter, refer to entry *name* in section N of the *UniPlus<sup>+</sup> User Manual*.

**Section 1** (*System Maintenance Commands and Application Programs*) contains system maintenance programs such as *fsck*, *mkfs*, etc., which generally reside in the directory */etc*; these entries carry a subsection designation of "M".

**Section 7** (*Special Files*) discusses the characteristics of each system file that actually refers to an input/output device. The names in this section generally refer to device names for the hardware, rather than to the names of the special files themselves.

**Section 8** (*System Maintenance Procedures*) discusses crash recovery and boot procedures.

Each section consists of several entries, each a page or so long. The name of the entry appears in the upper corners of its pages. Entries within each section are alphabetized, except the introduction that begins each section. The page numbers of each entry start at 1. Some entries may describe several routines, commands, etc. In such cases, the entry appears only once, alphabetized under its "major" name.

## INTRODUCTION

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

**NAME** gives the name(s) and a brief description of the entry.

**SYNOPSIS** summarizes the use of the program. A few conventions are used, particularly in Section 1 (*Commands*):

**Boldface** strings are typed just as they appear.

*Italic* strings usually represent substitutable argument prototypes (such as *filename*) which you are expected to substitute for the actual name. When an argument prototype is given as "name" or "file", it always refers to a *file* name.

Square brackets [] around an argument prototype indicate that the argument is optional.

Ellipses ... show that the previous argument prototype may be repeated.

A final convention is used by the commands themselves. An argument beginning with a minus -, plus +, or equal sign = is often taken to a flag argument, even if it appears in a position where a file name could appear. Therefore, it is unwise to have files whose names begin with -, +, or =.

**DESCRIPTION** discusses the program.

**EXAMPLE(S)** gives example(s) of usage.

**FILES** gives the file names that are built into the program.

**SEE ALSO** gives pointers to related information.

## INTRODUCTION

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

**WARNINGS** points out potential pitfalls.

**BUGS** gives known bugs and sometimes deficiencies. Occasionally, the suggested fix is also described.

At the front of each volume there is a table of contents and a permuted index. The permuted index lists the commands by the information in the NAME part of each entry in the User and Administrator Manual. The permuted index contains three columns. The center column is an alphabetic list of keywords. The last column is the entry that the keyword in the center column refers to. This entry is followed by the appropriate section number in parentheses. The first column contains the remaining information from the NAME part that either precedes or follows the keyword.

For example, to look for a text editor, scan the center column for the word "editor." There are several index lines containing an "editor reference, i.e.:

```
ed, red: text      editor      ..... ed(1)
files. ld: link    editor for common object ..... ld(1)
```

You can then turn to the entries listed in the last column, *ed(1)* and *ld(1)*, to find information on the editor.

On most systems, all entries are available on-line via the *man(1)* command.



## TABLE OF CONTENTS

### 1. System Maintenance Commands and Application Programs

intro	introduction to system maintenance commands and application programs
accept	allow/prevent LP requests
acct	overview of accounting and miscellaneous accounting commands
acctms	command summary from per-process accounting records
accton	connect-time accounting
acctmerg	merge or add total accounting files
acctprc	process accounting
acctsh	shell procedures for accounting
badblk	program to set or update bad block information
bcopy	interactive block copy
brc	system initialization shell scripts
checkall	faster file system checking procedure
chgnod	change current UNIX system nodename
chroot	change root directory for a command
clri	clear i-node
config	configure system
cpset	install object files in binary directories
cron	clock daemon
dcopy	copy file systems for optimal access time
devnm	device name
df	report number of free disk blocks
diskformat	format a disk
disktune	tune floppy disk settling time parameters
diskusg	generate disk accounting data by user ID
errdead	extract error records from dump
errdemon	error-logging daemon
errpt	process a report of logged errors
errstop	terminate the error-logging daemon
ff	list file names and statistics for a file system
filesave	daily/weekly UNIX file system backup
finc	fast incremental backup
frec	recover files from a backup tape
fsck	file system consistency check and interactive repair
fscv	convert files between M68000 and VAX-11/780 processors
fsdb	file system debugger
fuser	identify processes using a file or file structure
fwtmp	manipulate connect accounting records
getty	set terminal type, modes, speed, and line discipline
init	process control initialization
install	install commands
killall	kill all active processes
link	exercise link and unlink system calls
lpadmin	configure the LP spooling system
lpsched	start/stop the LP request scheduler and move requests
mkfs	construct a file system
mkfslb	construct a file system
mklost+found	make a lost+found directory for fsck
mknod	build special file
mount	mount and dismount file system
mvdir	move a directory
ncheck	generate names from i-numbers
pstat	print system facts
pwck	password/group file checkers

## Table of Contents

reboot . . . . .	reboot the system
runacct . . . . .	run daily accounting
sar . . . . .	system activity report package
setmnt . . . . .	establish mount table
shutdown . . . . .	terminate all processing
sysdef . . . . .	system definition
tic . . . . .	terminfo compiler
updater . . . . .	update files between two machines
uuclean . . . . .	uucp spool directory clean-up
uusub . . . . .	monitor uucp network
vchk . . . . .	version checkup
volcopy . . . . .	copy file systems with label checking
wall . . . . .	write to all users
whodo . . . . .	who is doing what

### 7. Special Files

intro . . . . .	introduction to special files
aliases . . . . .	aliases file for delivermail
error . . . . .	error-logging interface
mem . . . . .	core memory
null . . . . .	the null file
sxt . . . . .	pseudo-device driver
termio . . . . .	general terminal interface
tty . . . . .	controlling terminal interface

### 8. System Maintenance Procedures

intro . . . . .	introduction to system maintenance procedures
boot . . . . .	startup procedures
crash . . . . .	what to do when the system crashes
delivermail . . . . .	deliver mail to arbitrary people
ftpd . . . . .	DARPA Internet File Transfer Protocol server
ifconfig . . . . .	configure network interface parameters
netmail . . . . .	the B-NET network mail system
netmailer . . . . .	deliver mail to B-NET
remshd . . . . .	remote shell server
rexecd . . . . .	remote execution server
rlogind . . . . .	remote login server
route . . . . .	manually manipulate the routing tables
routed . . . . .	network routing daemon
rwhod . . . . .	system status server
telnetd . . . . .	DARPA TELNET protocol server
tftpd . . . . .	DARPA Trivial File Transfer Protocol server
trpt . . . . .	transliterate protocol trace



## COMMANDS

**300(1)** .....handle special functions of DASI 300 terminal  
**300s (See 300(1))** .....handle special functions of DASI 300s terminal  
**4014(1)** .....paginator for the Tektronix 4014 terminal  
**450(1)** .....handle special functions of the DASI 450 terminal  
**\_exit (See exit(2))** .....terminate process  
**\_tolower (See conv(3C))** .....translate characters  
**\_toupper (See conv(3C))** .....translate characters  
**a.out(4)** .....common assembler and link editor output  
**a.out5.0(4)** .....assembler and link editor output (System V a.out format only)  
**a64l(3C)** .....convert between long integer and base-64 ASCII string  
**abort(3C)** .....generate an IOT fault  
**abort(3F)** .....terminate Fortran program  
**abs(3C)** .....return integer absolute value  
**abs(3F)** .....Fortran absolute value  
**accept(1M)** .....allow LP requests  
**accept(2N)** .....accept a connection on a socket  
**access(2)** .....determine accessibility of a file  
**acct(1M)** .....overview of accounting  
**acct(2)** .....enable or disable process accounting  
**acct(4)** .....per-process accounting file format  
**acctens(1M)** .....command summary from per-process accounting records  
**acctcom(1)** .....search and print process accounting file(s)  
**acctcom(1M)** .....connect-time accounting  
**acctcom1 (See acctcom(1M))** .....connect-time accounting  
**acctcom2 (See acctcom(1M))** .....connect-time accounting  
**acctdisk (See acct(1M))** .....miscellaneous accounting command  
**acctdusg (See acct(1M))** .....miscellaneous accounting command  
**acctmerg(1M)** .....merge or add total accounting files  
**accton (See acct(1M))** .....miscellaneous accounting command  
**acctpre(1M)** .....process accounting  
**acctpre1 (See acctpre(1M))** .....process accounting  
**acctpre2 (See acctpre(1M))** .....process accounting  
**acctsh(1M)** .....shell procedures for accounting  
**acctwtmp (See acct(1M))** .....miscellaneous accounting command  
**acos (See trig(3M))** .....trigonometric function  
**acos(3F)** .....Fortran arccosine intrinsic function  
**admin(1)** .....create and administer SCCS files

## COMMANDS

**adventure(6)** ..... an exploration game  
**almag(3F)** ..... Fortran imaginary part of complex argument  
**aint(3F)** ..... Fortran integer part intrinsic function  
**alarm(2)** ..... set a process's alarm clock  
**aliases(7N)** ..... aliases file for delivermail  
**aliens(6)** ..... the alien invaders attack the earth  
**alog (See log(3F))** ..... Fortran natural logarithm intrinsic function  
**alog10 (See log10(3F))** ..... Fortran common logarithm intrinsic function  
**altblk(4)** ..... alternate block information for bad block handling  
**amax0 (See max(3F))** ..... Fortran maximum-value function  
**amax1 (See max(3F))** ..... Fortran maximum-value function  
**amin0 (See min(3F))** ..... Fortran minimum-value function  
**amin1 (See min(3F))** ..... Fortran minimum-value function  
**amod (See mod(3F))** ..... Fortran remaindering intrinsic function  
**and (See bool(3F))** ..... Fortran bitwise boolean function  
**anInt (See round(3F))** ..... Fortran nearest integer function  
**aouthdr(4)** ..... a.out header for common object files  
**ar(1)** ..... archive and library maintainer for portable archives  
**ar(4)** ..... common archive file format  
**ar5.0(1)** ..... archive and library maintainer (System V a.out format only)  
**ar5.0(4)** ..... archive (library) file format (System V a.out format only)  
**arithmetic(6)** ..... provide drill in number facts  
**arp(5P)** ..... Address Resolution Protocol  
**as(1)** ..... common assembler  
**as5.0(1)** ..... assembler (System V a.out format only)  
**asa(1)** ..... interpret ASA carriage control characters  
**ascii(5)** ..... map of ASCII character set  
**asetime (See ctime(3C))** ..... convert date and time to string  
**asin (See trig(3M))** ..... trigonometric function  
**asin(3F)** ..... Fortran arcsine intrinsic function  
**assert(3X)** ..... verify program assertion  
**at(1)** ..... execute commands at a later time  
**atan (See trig(3M))** ..... trigonometric function  
**atan(3F)** ..... Fortran arctangent intrinsic function  
**atan2 (See trig(3M))** ..... trigonometric function  
**atan2(3F)** ..... Fortran arctangent intrinsic function  
**atof(3C)** ..... convert ASCII string to floating-point number  
**atoi (See strtol(3C))** ..... convert string to integer  
**atol (See strtol(3C))** ..... convert string to integer

## COMMANDS

**autorobots(6)** .....escape from the automatic robots  
**awk(1)** .....pattern scanning and processing language  
**back(6)** .....the game of backgammon  
**badblk(1M)** .....program to set or update bad block information  
**banner(1)** .....make posters  
**banner7(1)** .....print large banner on printer  
**basename(1)** .....deliver portions of path names  
**batch (See at(1))** .....execute commands at a later time  
**bc(1)** .....arbitrary-precision arithmetic language  
**bed(6)** .....convert to antique media  
**bcheckrc (See brc(1M))** .....system initialization shell script  
**bcmp (See bstring(3N))** .....byte string operation  
**bcopy (See bstring(3N))** .....byte string operation  
**bcopy(1M)** .....interactive block copy  
**bdiff(1)** .....big diff  
**bessel(3M)** .....Bessel functions  
**bfs(1)** .....big file scanner  
**bind(2N)** .....bind a name to a socket  
**bj(6)** .....the game of black jack  
**blt(3C)** .....block transfer data  
**blt512 (See blt(3C))** .....block transfer data  
**bool(3F)** .....Fortran bitwise boolean functions  
**boot(8)** .....startup procedures  
**brc(1M)** .....system initialization shell script  
**brk(2)** .....change data segment space allocation  
**bs(1)** .....a compiler/interpreter for modest-sized programs  
**bsearch(3C)** .....binary search a sorted table  
**bstring(3N)** .....bit and byte string operations  
**byteorder(3N)** .....convert values between host and network byte order  
**bzero (See bstring(3N))** .....byte string operation  
**fabs (See abs(3F))** .....Fortran absolute value  
**cal(1)** .....print calendar  
**calendar(1)** .....reminder service  
**calloc (See malloc(3C))** .....main memory allocator  
**ccalloc (See malloc(3X))** .....fast main memory allocator  
**cancel (See lp(1))** .....cancel requests to an LP line printer  
**cat(1)** .....concatenate and print files  
**cb(1)** .....C program beautifier  
**cc(1)** .....C compiler

## COMMANDS

**cc5.0(1)** ..... C compiler (System V a.out format only)  
**ccos** (See *cos(3F)*) ..... Fortran cosine intrinsic function  
**cd(1)** ..... change working directory  
**cdc(1)** ..... change the delta commentary of an SCCS delta  
**cell** (See *floor(3M)*) ..... ceiling function  
**cexp** (See *exp(3F)*) ..... Fortran exponential intrinsic function  
**cflow(1)** ..... generate C flowgraph  
**char** (See *fype(3F)*) ..... explicit Fortran type conversion  
**chargefee** (See *acctsh(1M)*) ..... shell procedure for accounting  
**chase(6)** ..... try to escape the killer robots  
**chdir(2)** ..... change working directory  
**checkall(1M)** ..... faster file system checking procedure  
**checkw** (See *cw(1)*) ..... check text prepared with CW commands  
**checkeq** (See *eqn(1)*) ..... check text prepared with eqn or neqn commands  
**checklist(4)** ..... list of file systems processed by fsck  
**checkmm** (See *mm(1)*) ..... check documents formatted with the MM macros  
**chgnod(1M)** ..... change current UNIX system nodename  
**chgrp** (See *chown(1)*) ..... change group  
**chmod(1)** ..... change mode  
**chmod(2)** ..... change mode of file  
**chown(1)** ..... change owner  
**chown(2)** ..... change owner and group of a file  
**chroot(1M)** ..... change root directory for a command  
**chroot(2)** ..... change root directory  
**ckpacct** (See *acctsh(1M)*) ..... shell procedure for accounting  
**clear(1)** ..... clear terminal screen  
**clearerr** (See *ferroc(3S)*) ..... stream status inquiry  
**clock(3C)** ..... report CPU time used  
**clog** (See *log(3F)*) ..... Fortran natural logarithm intrinsic function  
**close(2)** ..... close a file descriptor  
**closedir** (See *directory(3X)*) ..... flexible length directory operation  
**clri(1M)** ..... clear inode  
**cmp(1)** ..... compare two files  
**cmplx** (See *fype(3F)*) ..... explicit Fortran type conversion  
**col(1)** ..... filter reverse line-feeds  
**comb(1)** ..... combine SCCS deltas  
**comm(1)** ..... select or reject lines common to two sorted files  
**config(1M)** ..... configure system  
**conj(3F)** ..... Fortran complex conjugate intrinsic function

## COMMANDS

**connect(2N)** ..... initiate a connection on a socket  
**conv(1)** ..... object file converter  
**conv(3C)** ..... translate characters  
**core(4)** ..... format of core image file  
**cos** (*See trig(3M)*) ..... trigonometric function  
**cos(3F)** ..... Fortran cosine intrinsic function  
**cosh** (*See sinh(3M)*) ..... hyperbolic function  
**cosh(3F)** ..... Fortran hyperbolic cosine intrinsic function  
**cp(1)** ..... copy files  
**cpio(1)** ..... copy file archives in and out  
**cpio(4)** ..... format of cpio archive  
**cpp(1)** ..... the C language preprocessor  
**cpp5.0(1)** ..... the C language preprocessor (System V a.out format only)  
**cpset(1M)** ..... install object files in binary directories  
**craps(6)** ..... the game of craps  
**crash(8)** ..... what to do when the system crashes  
**creat(2)** ..... create a new file or rewrite an existing one  
**cribbage(6)** ..... the card game cribbage  
**cron(1M)** ..... clock daemon  
**crontab(1)** ..... user crontab file  
**crypt(1)** ..... encode/decode  
**crypt(3C)** ..... generate DES encryption  
**csb(1)** ..... a shell (command interpreter) with C-like syntax  
**csin** (*See sin(3F)*) ..... Fortran sine intrinsic function  
**csplit(1)** ..... context split  
**csqrt** (*See sqrt(3F)*) ..... Fortran square root intrinsic function  
**ct(1C)** ..... spawn getty to a remote terminal  
**ctags(1)** ..... maintain a tags file for a C program  
**ctermid(3S)** ..... generate filename for terminal  
**ctime(3C)** ..... convert date and time to string  
**ctrace(1)** ..... C program debugger  
**ctype(3C)** ..... classify characters  
**cu(1C)** ..... call another UNIX system  
**cubic** (*See m(6)*) ..... tic-tac-toe  
**curses(3X)** ..... CRT screen handling and optimization package  
**cuserid(3S)** ..... get character login name of the user  
**cut(1)** ..... cut out selected fields of each line of a file  
**cw(1)** ..... prepare constant-width text for troff  
**cxref(1)** ..... generate C program cross-reference

## COMMANDS

<b>dabs</b> (See <i>abs(3F)</i> )	Fortran absolute value
<b>dacos</b> (See <i>acos(3F)</i> )	Fortran arccosine intrinsic function
<b>dasin</b> (See <i>asin(3F)</i> )	Fortran arcsine intrinsic function
<b>datan</b> (See <i>atan(3F)</i> )	Fortran arctangent intrinsic function
<b>datan2</b> (See <i>atan2(3F)</i> )	Fortran arctangent intrinsic function
<b>date(1)</b>	print and set the date
<b>dbie</b> (See <i>ftype(3F)</i> )	explicit Fortran type conversion
<b>dc(1)</b>	desk calculator
<b>dcmplx</b> (See <i>ftype(3F)</i> )	explicit Fortran type conversion
<b>dconjg</b> (See <i>conjg(3F)</i> )	Fortran complex conjugate intrinsic function
<b>dcopy(1M)</b>	copy file systems for optimal access time
<b>dcopy1b(1M)</b>	copy file systems for optimal access time
<b>dcos</b> (See <i>cos(3F)</i> )	Fortran cosine intrinsic function
<b>dcosh</b> (See <i>cosh(3F)</i> )	Fortran hyperbolic cosine intrinsic function
<b>dd(1)</b>	convert and copy a file
<b>ddim</b> (See <i>dim(3F)</i> )	positive difference intrinsic function
<b>delivermail(8N)</b>	deliver mail to arbitrary people
<b>delta(1)</b>	make a delta (change) to an SCCS file
<b>deroff(1)</b>	remove <b>broff/troff</b> , <b>tbl</b> , and <b>eqn</b> constructs
<b>dexp</b> (See <i>exp(3F)</i> )	Fortran exponential intrinsic function
<b>devnam(1M)</b>	device name
<b>df(1M)</b>	report number of free disk blocks
<b>dfsek</b> (See <i>fsck(1M)</i> )	file system consistency check and interactive repair
<b>dlal(3C)</b>	establish an out-going terminal line connection
<b>diff(1)</b>	differential file comparator
<b>diff3(1)</b>	3-way differential file comparison
<b>diffdir(1)</b>	diff directories
<b>diffmk(1)</b>	mark differences between files
<b>dim(3F)</b>	positive difference intrinsic functions
<b>dimag</b> (See <i>aimag(3F)</i> )	Fortran imaginary part of complex argument
<b>dint</b> (See <i>atn(3F)</i> )	Fortran integer part intrinsic function
<b>dir(4)</b>	format of directories
<b>dircmp(1)</b>	directory comparison
<b>directory(3X)</b>	flexible length directory operations
<b>dirname</b> (See <i>basename(1)</i> )	deliver portions of path names
<b>dis(1)</b>	disassembler
<b>disable</b> (See <i>enable(1)</i> )	disable LP printers
<b>diskformat(1M)</b>	format a disk
<b>disktune(1M)</b>	tune floppy disk settling time parameters

## COMMANDS

**diskusg(1M)** .....generate disk accounting data by user ID  
**dlog** (See *log(3F)*) .....Fortran natural logarithm intrinsic function  
**dlog10** (See *log10(3F)*) .....Fortran common logarithm intrinsic function  
**dmax1** (See *max(3F)*) .....Fortran maximum-value function  
**dmin1** (See *min(3F)*) .....Fortran minimum-value function  
**dmod** (See *mod(3F)*) .....Fortran remaindering intrinsic function  
**drint** (See *round(3F)*) .....Fortran nearest integer function  
**dodisk** (See *acctsh(1M)*) .....shell procedure for accounting  
**dprod(3F)** .....double precision product intrinsic function  
**drand48(3C)** .....generate uniformly distributed pseudo-random numbers  
**dsign** (See *sign(3F)*) .....Fortran transfer-of-sign intrinsic function  
**dsin** (See *sin(3F)*) .....Fortran sine intrinsic function  
**dsinh** (See *sinh(3F)*) .....Fortran hyperbolic sine intrinsic function  
**dsqrt** (See *sqrt(3F)*) .....Fortran square root intrinsic function  
**dtan** (See *tan(3F)*) .....Fortran tangent intrinsic function  
**dtanh** (See *tanh(3F)*) .....Fortran hyperbolic tangent intrinsic function  
**du(1)** .....summarize disk usage  
**dump(1)** .....dump selected parts of an object file  
**dup(3)** .....duplicate a descriptor  
**dup2(3N)** .....duplicate a descriptor  
**echo(1)** .....echo arguments  
**ecvt(3C)** .....convert floating-point number to string  
**ed(1)** .....text editor  
**edata** (See *end(3C)*) .....last locations in program  
**edit** (See *ex(1)*) .....text editor  
**efl(1)** .....Extended Fortran Language  
**egrep** (See *grep(1)*) .....search a file for a pattern  
**enable(1)** .....enable LP printers  
**encrypt** (See *crypt(3C)*) .....generate DES encryption  
**end(3C)** .....last locations in program  
**endgrent** (See *getgrent(3C)*) .....obtain group file entry from a group file  
**endhostent** (See *gethostent(3N)*) .....get network host entry  
**endnetent** (See *getnetent(3N)*) .....get network entry  
**endprotoent** (See *getprotoent(3N)*) .....get protocol entry  
**endpwent** (See *getpwent(3C)*) .....get password file entry  
**endservent** (See *getservent(3N)*) .....get service entry  
**endutent** (See *getut(3C)*) .....access utmp file entry  
**env(1)** .....set environment for command execution  
**environ(5)** .....user environment

## COMMANDS

**eqn**(1) .....format mathematical text for troff  
**eqschar**(5) .....special character definitions for eqn and neqn  
**erand48** (See *drand48(3C)*) .....generate uniformly distributed pseudo-random numbers  
**erf**(3M) .....error function  
**erfc** (See *erf(3M)*) .....complementary error function  
**errdead**(1M) .....extract error records from dump  
**errdemon**(1M) .....error-logging daemon  
**errfile**(4) .....error-log file format  
**errno** (See *perror(3C)*) .....system error message  
**error**(7) .....error-logging interface  
**errpt**(1M) .....process a report of logged errors  
**errstop**(1M) .....terminate the error-logging daemon  
**etext** (See *end(3C)*) .....last locations in program  
**ex**(1) .....text editor  
**exec**(2) .....execute a file  
**exec1** (See *exec(2)*) .....execute a file  
**execle** (See *exec(2)*) .....execute a file  
**execlp** (See *exec(2)*) .....execute a file  
**execv** (See *exec(2)*) .....execute a file  
**execve** (See *exec(2)*) .....execute a file  
**execvp** (See *exec(2)*) .....execute a file  
**exit**(2) .....terminate process  
**exp**(3F) .....Fortran exponential intrinsic function  
**exp**(3M) .....exponential function  
**expr**(1) .....evaluate arguments as an expression  
**exterr**(1) .....turn on/off the extended errors in the specified device  
**f77**(1) .....Fortran 77 compiler  
**fabs** (See *floor(3M)*) .....absolute value function  
**factor**(1) .....factor a number  
**false** (See *true(1)*) .....provide truth values  
**fclose**(3S) .....close a stream  
**fcntl**(2) .....file control  
**fcntl**(5) .....file control options  
**fevt** (See *ecvt(3C)*) .....convert floating-point number to string  
**fdopen** (See *fdopen(3S)*) .....open a stream  
**feof** (See *ferror(3S)*) .....stream status inquiry  
**ferror**(3S) .....stream status inquiry  
**ff**(1M) .....list file names and statistics for a file system  
**flush** (See *fclose(3S)*) .....flush a stream



## COMMANDS

**fts** (See *bstring(3N)*) ..... bit string operation

**getc** (See *getc(3S)*) ..... get character from a stream

**getgrent** (See *getgrent(3C)*) ..... obtain group file entry from a group file

**getpwent** (See *getpwent(3C)*) ..... get password file entry

**gets** (See *gets(3S)*) ..... get a string from a stream

**grep** (See *grep(1)*) ..... search a file for a pattern

**file(1)** ..... determine file type

**filehdr(4)** ..... file header for common object files

**fileso** (See *ferror(3S)*) ..... stream status inquiry

**filesave(1M)** ..... daily/weekly UNIX file system backup

**fnac(1M)** ..... fast incremental backup

**find(1)** ..... find files

**fish(6)** ..... play "Go Fish"

**float** (See *ftype(3F)*) ..... explicit Fortran type conversion

**floor(3M)** ..... floor function

**fmod** (See *floor(3M)*) ..... remainder function

**fopen(3S)** ..... open a stream

**fork(2)** ..... create a new process

**fortune(6)** ..... print a random, hopefully interesting, adage

**fprintf** (See *printf(3S)*) ..... print formatted output

**fputc** (See *putc(3S)*) ..... put character on a stream

**fputs** (See *puts(3S)*) ..... put a string on a stream

**fread(3S)** ..... binary input

**frec(1M)** ..... recover files from a backup tape

**free** (See *malloc(3C)*) ..... main memory allocator

**free** (See *malloc(3X)*) ..... fast main memory allocator

**freopen** (See *fopen(3S)*) ..... open a stream

**freq(1)** ..... report on character frequencies in a file

**frexp(3C)** ..... manipulate parts of floating-point numbers

**fs(4)** ..... format of system volume

**fscanf** (See *scanf(3S)*) ..... convert formatted input

**fsck(1M)** ..... file system consistency check and interactive repair

**fsconv(1M)** ..... convert files between M68000 and VAX-11/780 processors

**fsdb(1M)** ..... file system debugger

**fseek(3S)** ..... reposition a file pointer in a stream

**fspec(4)** ..... format specification in text files

**fsplit(1)** ..... split f77, ratfor, or efi files

**fstat** (See *stat(2)*) ..... get file status

**ftell** (See *fseek(3S)*) ..... reposition a file pointer in a stream

## COMMANDS

<i>ftok</i> (See <i>stipc(3C)</i> )	.....	standard interprocess communication package
<i>ftp</i> (1N)	.....	file transfer program
<i>ftpd</i> (8N)	.....	DARPA Internet File Transfer Protocol server
<i>ftw</i> (3C)	.....	walk a file tree
<i>ftype</i> (3F)	.....	explicit Fortran type conversion
<i>fuser</i> (1M)	.....	identify processes using a file or file structure
<i>fwrite</i> (See <i>fread(3S)</i> )	.....	binary output
<i>fwtmp</i> (1M)	.....	manipulate connect accounting records
<i>gamma</i> (3M)	.....	log gamma function
<i>gcvl</i> (See <i>ecvl(3C)</i> )	.....	convert floating-point number to string
<i>get</i> (1)	.....	get a version of an SCCS file
<i>getarg</i> (3F)	.....	return Fortran command-line argument
<i>getc</i> (3S)	.....	get character from a stream
<i>getchar</i> (See <i>getc(3S)</i> )	.....	get character from a stream
<i>getcwd</i> (3C)	.....	get pathname of current working directory
<i>getdtablesize</i> (3N)	.....	get descriptor table size
<i>getegid</i> (See <i>getuid(2)</i> )	.....	get effective group ID
<i>getenv</i> (3C)	.....	return value for environment name
<i>getenv</i> (3F)	.....	return Fortran environment variable
<i>geteuid</i> (See <i>getuid(2)</i> )	.....	get effective user ID
<i>getgid</i> (See <i>getuid(2)</i> )	.....	get real group ID
<i>getgrent</i> (3C)	.....	obtain group file entry from a group file
<i>getgrgid</i> (See <i>getgrent(3C)</i> )	.....	obtain group file entry from a group file
<i>getgrnam</i> (See <i>getgrent(3C)</i> )	.....	obtain group file entry from a group file
<i>gethostbyaddr</i> (See <i>gethostent(3N)</i> )	.....	get network host entry
<i>gethostbyname</i> (See <i>gethostent(3N)</i> )	.....	get network host entry
<i>gethostent</i> (3N)	.....	get network host entry
<i>gethostid</i> (2N)	.....	get unique identifier of current host
<i>gethostname</i> (2N)	.....	get name of current host
<i>getlogin</i> (3C)	.....	get login name
<i>getnetbyaddr</i> (See <i>getnetent(3N)</i> )	.....	get network entry
<i>getnetbyname</i> (See <i>getnetent(3N)</i> )	.....	get network entry
<i>getnetent</i> (3N)	.....	get network entry
<i>getopt</i> (1)	.....	parse command options
<i>getopt</i> (3C)	.....	get option letter from argument vector
<i>getpass</i> (3C)	.....	read a password
<i>getpeername</i> (2N)	.....	get name of connected peer
<i>getpgid</i> (See <i>getpid(2)</i> )	.....	get process group ID
<i>getpid</i> (2)	.....	get process ID

## COMMANDS

**getppid** (*See getpid(2)*) .....get parent process ID  
**getprotobyname** (*See getprotoent(3N)*) .....get protocol entry  
**getprotobynumber** (*See getprotoent(3N)*) .....get protocol entry  
**getprotoent(3N)** .....get protocol entry  
**getpw(3C)** .....get name from UID  
**getpwent(3C)** .....get password file entry  
**getpwnam** (*See getpwent(3C)*) .....get password file entry  
**getpwuid** (*See getpwent(3C)*) .....get password file entry  
**gets(3S)** .....get a string from a stream  
**getservbyname** (*See getservent(3N)*) .....get service entry  
**getservbyport** (*See getservent(3N)*) .....get service entry  
**getservent(3N)** .....get service entry  
**getsockname(2N)** .....get socket name  
**getsockopt(2N)** .....get options on sockets  
**getty(1M)** .....set terminal type, modes, speed, and line discipline  
**gettydefs(4)** .....speed and terminal settings used by getty  
**getuid(2)** .....get real user ID  
**getut(3C)** .....access utmp file entry  
**getutent** (*See getut(3C)*) .....access utmp file entry  
**getutid** (*See getut(3C)*) .....access utmp file entry  
**getutline** (*See getut(3C)*) .....access utmp file entry  
**getw** (*See getc(3S)*) .....get word from a stream  
**gmtime** (*See ctime(3C)*) .....convert date and time to string  
**graph(1G)** .....draw a graph  
**greek(1)** .....select terminal filter  
**greek(5)** .....graphics for the extended TTY-37 type-box  
**grep(1)** .....search a file for a pattern  
**group(4)** .....group file  
**grpck** (*See pwck(1M)*) .....group file checker  
**gsignal** (*See ssignal(3C)*) .....software signal  
**hangman(6)** .....guess the word  
**hashcheck** (*See spell(1)*) .....work with the **spell** program's hash lists  
**hashmake spell(1)** .....work with the **spell** program's hash lists  
**hcreate** (*See hsearch(3C)*) .....manage hash search tables  
**hdestroy** (*See hsearch(3C)*) .....manage hash search tables  
**head(1)** .....give first few lines  
**help(1)** .....ask for help in using SCCS  
**hex(1)** .....translates object files  
**hostid(1N)** .....set or print identifier of current host system

## COMMANDS

**hostname(1N)** .....set or print name of current host system  
**hosts(4N)** .....host name data base  
**hsearch(3C)** .....manage hash search tables  
**htonl (See byteorder(3N))** .....convert values between host and network byte order  
**htons (See byteorder(3N))** .....convert values between host and network byte order  
**hyphen(1)** .....find hyphenated words  
**hypot(3M)** .....Euclidean distance function  
**labs (See abs(3F))** .....Fortran absolute value  
**large(3F)** .....count command line arguments  
**lchar (See ftype(3F))** .....explicit Fortran type conversion  
**ld(1)** .....print user and group IDs and names  
**ldim (See dim(3F))** .....positive difference intrinsic function  
**ldint (See ftype(3F))** .....explicit Fortran type conversion  
**ldnint (See round(3F))** .....Fortran nearest integer function  
**ifconfig(8N)** .....configure network interface parameters  
**iflx (See ftype(3F))** .....explicit Fortran type conversion  
**index(3F)** .....return location of Fortran substring  
**inet(3N)** .....Internet address manipulation routines  
**inet(5F)** .....Internet protocol family  
**inet\_addr (See inet(3N))** .....Internet address manipulation routine  
**inet\_inof (See inet(3N))** .....Internet address manipulation routine  
**inet\_makaddr (See inet(3N))** .....Internet address manipulation routine  
**inet\_netof (See inet(3N))** .....Internet address manipulation routine  
**inet\_network (See inet(3N))** .....Internet address manipulation routine  
**inet\_ntoa (See inet(3N))** .....Internet address manipulation routine  
**init(1M)** .....process control initialization  
**inittab(4)** .....script for the init process  
**inode(4)** .....format of an inode  
**insque(3N)** .....insert element from a queue  
**install(1M)** .....install commands  
**int (See ftype(3F))** .....explicit Fortran type conversion  
**ioctl(2)** .....control device  
**ip(5P)** .....Internet Protocol  
**iperm(1)** .....remove a message queue, semaphore set or shared memory id  
**ipcs(1)** .....report inter-process communication facilities status  
**irand (See rand(3F))** .....Fortran uniform random-number generator  
**isalnum (See ctype(3C))** .....classify characters  
**isalpha (See ctype(3C))** .....classify characters  
**isascii (See ctype(3C))** .....classify characters

## COMMANDS

**isatty** (*See ttyname(3C)*) ..... find name of a terminal  
**iscntrl** (*See ctype(3C)*) ..... classify characters  
**isdigit** (*See ctype(3C)*) ..... classify characters  
**isgraph** (*See ctype(3C)*) ..... classify characters  
**isign** (*See sign(3F)*) ..... Fortran transfer-of-sign intrinsic function  
**islower** (*See ctype(3C)*) ..... classify characters  
**isprint** (*See ctype(3C)*) ..... classify characters  
**ispunct** (*See ctype(3C)*) ..... classify characters  
**isspace** (*See ctype(3C)*) ..... classify characters  
**issue(4)** ..... issue identification file  
**isupper** (*See ctype(3C)*) ..... classify characters  
**isxdigit** (*See ctype(3C)*) ..... classify characters  
**j0** (*See bessel(3M)*) ..... Bessel function  
**j1** (*See bessel(3M)*) ..... Bessel function  
**jn** (*See bessel(3M)*) ..... Bessel function  
**join(1)** ..... relational database operator  
**rand48** (*See drand48(3C)*) ..... generate uniformly distributed pseudo-random numbers  
**kill(1)** ..... terminate a process  
**kill(2)** ..... send a signal to a process or a group of processes  
**killall(1M)** ..... kill all active processes  
**killpg(3N)** ..... send signal to a process group  
**kmem** (*See mem(7)*) ..... core memory  
**l3tol(3C)** ..... convert between 3-byte integers and long integers  
**l64a** (*See a64l(3C)*) ..... convert between long integer and base-64 ASCII string  
**labelit** (*See volcopy(1M)*) ..... copy file systems with label checking  
**last(1)** ..... indicate last logins of users and teletypes  
**lastlogin** (*See acctsh(1M)*) ..... shell procedure for accounting  
**lav(1)** ..... print load average statistics  
**lcong48** (*See drand48(3C)*) ..... generate uniformly distributed pseudo-random numbers  
**ld(1)** ..... link editor for common object files  
**ld5.0(1)** ..... link editor (System V a.out format only)  
**ldclose** (*See ldclose(3X)*) ..... close a common object file  
**ldahread(3X)** ..... read the archive header of a member of an archive file  
**ldopen** (*See ldopen(3X)*) ..... open a common object file for reading  
**ldclose(3X)** ..... close a common object file  
**ldexp** (*See frexp(3C)*) ..... manipulate parts of floating-point numbers  
**ldfcn(4)** ..... common object file access routines  
**ldhread(3X)** ..... read the file header of a common object file  
**ldgetname(3X)** ..... retrieve symbol name for object file

## COMMANDS

**ldllnit** (See *ldtread(3X)*) ..manipulate line number entries of a common object file function  
**ldlitem** (See *ldtread(3X)*) ..manipulate line number entries of a common object file function  
**ldlread(3X)** .....manipulate line number entries of a common object file function  
**ldlseek(3X)** .....seek to line number entries of a section of a common object file  
**ldlnlseek** (See *ldlseek(3X)*) seek to line number entries of a section of a common object file  
**ldlnrseek** (See *ldrseek(3X)*) ..seek to relocation entries of a section of a common object file  
**ldlnshread** (See *ldshread(3X)*) ..read an indexed/named section header of a common object file  
**ldlnsseek** (See *ldsseek(3X)*) .....seek to an indexed/named section of a common object file  
**ldobseek(3X)** .....seek to the optional file header of a common object file  
**ldopen(3X)** .....open a common object file for reading  
**ldrseek(3X)** .....seek to relocation entries of a section of a common object file  
**ldshread(3X)** .....read an indexed/named section header of a common object file  
**ldsseek(3X)** .....seek to an indexed/named section of a common object file  
**ldtbindex(3X)** .....compute the index of a symbol table entry of a common object file  
**ldtbread(3X)** .....read an indexed symbol table entry of a common object file  
**ldtbseek(3X)** .....seek to the symbol table of a common object file  
**len(3F)** .....return length of Fortran string  
**lex(1)** .....generate programs for simple lexical tasks  
**lfind** (See *lsearch(3C)*) .....linear search and update  
**lge** (See *strcmp(3F)*) .....string comparison intrinsic function  
**lgt** (See *strcmp(3F)*) .....string comparison intrinsic function  
**llfe(6)** .....play the game of life  
**line(1)** .....read one line  
**llsenum(4)** .....line number entries in a common object file  
**link(1M)** .....exercise link system call  
**link(2)** .....link to a file  
**lint(1)** .....a C program checker  
**listen(2N)** .....listen for connections on a socket  
**lle** (See *strcmp(3F)*) .....string comparison intrinsic function  
**llt** (See *strcmp(3F)*) .....string comparison intrinsic function  
**ln** (See *cp(1)*) .....link files  
**lo(5)** .....software loopback network interface  
**localtime** (See *ctime(3C)*) .....convert date and time to string  
**lockf(3C)** .....record locking on files  
**locking(2)** .....provide exclusive file regions for reading or writing  
**log** (See *exp(3M)*) .....logarithm function  
**log(3F)** .....Fortran natural logarithm intrinsic function  
**log10** (See *exp(3M)*) .....logarithm function  
**log10(3F)** .....Fortran common logarithm intrinsic function

## COMMANDS

**login(1)** .....sign on  
**logname(1)** .....get login name  
**logname(3X)** .....return login name of user  
**longjmp** (See *setjmp(3C)*) .....non-local goto  
**lorder(1)** .....find ordering relation for an object library  
**lorder5.0(1)** .....find ordering relation for an object library (System V a.out format only)  
**lp(1)** .....send requests to an LP line printer  
**lpadmin(1M)** .....configure the LP spooling system  
**lpmove** (See *lpsched(1M)*) .....move LP requests  
**lpsched(1M)** .....start the LP request scheduler  
**lpshut** (See *lpsched(1M)*) .....stop the LP request scheduler  
**lpstat(1)** .....print LP status information  
**lrand48** (See *drand48(3C)*) .....generate uniformly distributed pseudo-random numbers  
**ls(1)** .....list contents of directory  
**lsearch(3C)** .....linear search and update  
**lseek(2)** .....move read/write file pointer  
**lshft** (See *bool(3F)*) .....Fortran bitwise boolean function  
**ltoi3** (See *ltoi3(3C)*) .....convert between 3-byte integers and long integers  
**m4(1)** .....macro processor  
**m68k** (See *machid(1)*) .....provide truth value about your processor type  
**machid(1)** .....provide truth value about your processor type  
**mail(1)** .....send mail to users or read mail  
**mailx(1)** .....interactive message processing system  
**make(1)** .....maintain, update, and regenerate groups of programs  
**makekey(1)** .....generate encryption key  
**mallinfo** (See *malloc(3X)*) .....fast main memory allocator  
**malloc(3C)** .....main memory allocator  
**malloc(3X)** .....fast main memory allocator  
**malloc** (See *malloc(3X)*) .....fast main memory allocator  
**man(1)** .....print entries in this manual  
**man(5)** .....macros for formatting entries in this manual  
**master(4)** .....master device information table  
**math(5)** .....math functions and constants  
**matherr(3M)** .....error-handling function  
**max(3F)** .....Fortran maximum-value function  
**max0** (See *max(3F)*) .....Fortran maximum-value function  
**max1** (See *max(3F)*) .....Fortran maximum-value function  
**maze(6)** .....generate a maze  
**mc68cc(1)** .....C compiler

## COMMANDS

**mclock(3F)** .....return Fortran time accounting  
**mem(7)** .....core memory  
**memcopy (See memory(3C))** .....memory operation  
**memchr (See memory(3C))** .....memory operation  
**memcmp (See memory(3C))** .....memory operation  
**memcpy (See memory(3C))** .....memory operation  
**memory(3C)** .....memory operation  
**memset (See memory(3C))** .....memory operation  
**msg(1)** .....permit or deny messages  
**min(3F)** .....Fortran minimum-value function  
**min0 (See min(3F))** .....Fortran minimum-value function  
**min1 (See min(3F))** .....Fortran minimum-value function  
**mkdir(1)** .....make a directory  
**mkfs(1M)** .....construct a file system  
**mkfs1b(1M)** .....construct a file system  
**mklost+found(1M)** .....make a lost+found directory for fsck  
**mknod(1M)** .....build special file  
**mknod(2)** .....make a directory, or a special or ordinary file  
**mkstr(1)** .....create an error message file by massaging C source  
**mktemp(3C)** .....make a unique filename  
**mm(1)** .....print documents formatted with the MM macros  
**mm(5)** .....the MM macro package for formatting documents  
**mmt(1)** .....typeset documents  
**mmtab(4)** .....mounted file system table  
**mod(3F)** .....Fortran remaindering intrinsic function  
**modf (See frexp(3C))** .....manipulate parts of floating-point numbers  
**monacct (See acctsh(1M))** .....shell procedure for accounting  
**monitor(3C)** .....prepare execution profile  
**moog(6)** .....guessing game  
**more(1)** .....file perusal filter for crt viewing  
**mosd(5)** .....the OSDD adapter macro package for formatting documents  
**mount(1M)** .....mount file system  
**mount(2)** .....mount a file system  
**mptx(5)** .....the macro package for formatting a permuted index  
**mrnd48 (See drand48(3C))** .....generate uniformly distributed pseudo-random numbers  
**msgctl(2)** .....message control operations  
**msgget(2)** .....get message queue  
**msgop(2)** .....message operations  
**mv (See cp(1))** .....move files



## COMMANDS

**mv(5)** ..... a troff macro package for typesetting view graphs and slides  
**mvdir(1M)** ..... move a directory  
**mvf (See mmt(1))** ..... typeset view graphs and slides  
**ncheck(1M)** ..... generate names from inumbers  
**neqn (See eqn(1))** ..... format mathematical text for nroff  
**netmail(8N)** ..... the B-NET network mail system  
**netmailer(8N)** ..... deliver mail to B-NET  
**netstat(1N)** ..... show network status  
**networks(4N)** ..... network name data base  
**newform(1)** ..... change the format of a text file  
**newgrp(1)** ..... log in to a new group  
**news(1)** ..... print news items  
**nice(1)** ..... run a command at low priority  
**nice(2)** ..... change priority of a process  
**nint (See round(3F))** ..... Fortran nearest integer function  
**nl(1)** ..... line numbering filter  
**nlst(3C)** ..... get entries from name list  
**nm(1)** ..... print name list of common object file  
**nm5.0(1)** ..... print name list (System V a.out format only)  
**nohup(1)** ..... run a command immune to hangups (sh only)  
**not (See bool(3F))** ..... Fortran bitwise boolean function  
**nrand48 (See drand48(3C))** ..... generate uniformly distributed pseudo-random numbers  
**nroff(1)** ..... format text  
**ntohl (See byteorder(3N))** ..... convert values between host and network byte order  
**ntohs (See byteorder(3N))** ..... convert values between host and network byte order  
**null(7)** ..... the null file  
**nulladm (See acctsh(1M))** ..... shell procedure for accounting  
**number(6)** ..... convert Arabic numerals to English  
**od(1)** ..... octal dump  
**open(2)** ..... open for reading or writing  
**opendir (See directory(3X))** ..... flexible length directory operation  
**or (See bool(3F))** ..... Fortran bitwise boolean function  
**osdd (See mm(1))** ..... print documents formatted with the MM and OSDD macros  
**pack(1)** ..... compress files  
**passwd(1)** ..... change login password  
**passwd(4)** ..... password file  
**paste(1)** ..... merge same lines of several files or subsequent lines of one file  
**pause(2)** ..... suspend process until signal  
**pcat (See pack(1))** ..... expand compressed files

## COMMANDS

<b>pclose</b> (See <i>popen(3S)</i> )	.....	initiate pipe to/from a process
<b>pdp11</b> (See <i>machid(1)</i> )	.....	provide truth value about your processor type
<b>perror(3C)</b>	.....	system error message
<b>pg(1)</b>	.....	file perusal filter for soft-copy terminals
<b>phys(2)</b>	.....	allow a process to access physical addresses
<b>pipe(2)</b>	.....	create an interprocess channel
<b>plck(2)</b>	.....	lock process, text, or data in memory
<b>plot(3X)</b>	.....	graphics interface subroutines
<b>plot(4)</b>	.....	graphics interface
<b>pnch(4)</b>	.....	file format for card images
<b>popen(3S)</b>	.....	initiate pipe to/from a process
<b>pow</b> (See <i>exp(3M)</i> )	.....	power function
<b>powerfall</b> (See <i>brc(1M)</i> )	.....	system initialization shell script
<b>pr(1)</b>	.....	print files
<b>prctmp</b> (See <i>acctsh(1M)</i> )	.....	shell procedure for accounting
<b>prdaily</b> (See <i>acctsh(1M)</i> )	.....	shell procedure for accounting
<b>printenv(1)</b>	.....	print out the environment
<b>printf(3S)</b>	.....	print formatted output
<b>prof(1)</b>	.....	display profile data
<b>prof(5)</b>	.....	profile within a function
<b>profil(2)</b>	.....	execution time profile
<b>profile(4)</b>	.....	setting up an environment at login time
<b>protocols(4N)</b>	.....	protocol name data base
<b>prs(1)</b>	.....	print an SCCS file
<b>prtacct</b> (See <i>acctsh(1M)</i> )	.....	shell procedure for accounting
<b>ps(1)</b>	.....	report process status
<b>psstat(1M)</b>	.....	print system facts
<b>ptrace(2)</b>	.....	process trace
<b>ptx(1)</b>	.....	permuted index
<b>pty(5)</b>	.....	pseudo terminal driver
<b>put(1C)</b>	.....	puts a file onto a remote machine
<b>putc(3S)</b>	.....	put character on a stream
<b>putchar</b> (See <i>putc(3S)</i> )	.....	put character on a stream
<b>putenv(3C)</b>	.....	change or add value to environment
<b>putpwent(3C)</b>	.....	write password file entry
<b>puts(3S)</b>	.....	put a string on a stream
<b>pututline</b> (See <i>getut(3C)</i> )	.....	access utmp file entry
<b>putw</b> (See <i>putc(3S)</i> )	.....	put word on a stream
<b>pwck(1M)</b>	.....	password file checker

## COMMANDS

**pwd(1)** .....working directory name  
**qsort(3C)** .....quicker sort  
**quiz(6)** .....test your knowledge  
**rain(6)** .....animated raindrops display  
**rand(3C)** .....simple random-number generator  
**rand(3F)** .....Fortran uniform random-number generator  
**ratfor(1)** .....rational Fortran dialect  
**rc (See *brc(1M)*)** .....system initialization shell script  
**rcmd(3N)** .....routine for returning a stream to a remote command  
**rcp(1N)** .....remote file copy  
**rcvhex(1)** .....translates Motorola S-records from downloading into a file  
**read(2)** .....read from file  
**readdir (See *directory(3X)*)** .....flexible length directory operation  
**readv(3N)** .....read from file  
**real (See *fype(3F)*)** .....explicit Fortran type conversion  
**realloc (See *malloc(3C)*)** .....main memory allocator  
**realloc (See *malloc(3X)*)** .....fast main memory allocator  
**reboot(1M)** .....reboot the system  
**reboot(2)** .....reboot the system  
**recv(2N)** .....receive a message from a socket  
**recvfrom (See *recv(2N)*)** .....receive a message from a socket  
**recvmsg (See *recv(2N)*)** .....receive a message from a socket  
**red (See *ed(1)*)** .....text editor  
**regcmp(1)** .....regular expression compile  
**regcmp(3X)** .....compile a regular expression  
**regex (See *regcmp(3X)*)** .....execute a regular expression  
**regexp(5)** .....regular expression compile and match routines  
**reject (See *accept(1M)*)** .....prevent LP requests  
**reloc(4)** .....relocation information for a common object file  
**remque (See *insque(3N)*)** .....remove element from a queue  
**remsh(1N)** .....remote shell  
**remshd(8N)** .....remote shell server  
**reset (See *iset(1)*)** .....reset the teletype bits to a sensible state  
**rewind (See *fseek(3S)*)** .....reposition a file pointer in a stream  
**rewinddir (See *directory(3X)*)** .....flexible length directory operation  
**rexec(3N)** .....return stream to a remote command  
**rexecd(8N)** .....remote execution server  
**rlogin(1N)** .....remote login  
**rlogind(8N)** .....remote login server

## COMMANDS

**rm(1)** .....remove files  
**rmail** (*See mail(1)*) .....send mail to users or read mail  
**rm-del(1)** .....remove a delta from an SCCS file  
**rmdir** (*See rm(1)*) .....remove directories  
**robots(6)** .....escape from the robots  
**round(3F)** .....Fortran nearest integer functions  
**route(8N)** .....manually manipulate the routing tables  
**routed(8N)** .....network routing daemon  
**rrsvport** (*See rcmd(3N)*) .....routine for returning a stream to a remote command  
**rsh** (*See sh(1)*) .....shell, the restricted command programming language  
**rshift** (*See bool(3F)*) .....Fortran bitwise boolean function  
**runacct(1M)** .....run daily accounting  
**ruptime(1N)** .....show host status of local machines  
**ruserok** (*See rcmd(3N)*) .....routine for returning a stream to a remote command  
**rwho(1N)** .....who's logged in on local machines  
**rwhod(8N)** .....system status server  
**sa1** (*See sar(1M)*) .....system activity report package  
**sa2** (*See sar(1M)*) .....system activity report package  
**sact(1)** .....print current SCCS file editing activity  
**sadc** (*See sar(1M)*) .....system activity report package  
**sag(1G)** .....system activity graph  
**sar(1)** .....system activity reporter  
**sar(1M)** .....system activity report package  
**sbrk** (*See brk(2)*) .....change data segment space allocation  
**scanf(3S)** .....convert formatted input  
**secsdiff(1)** .....compare two versions of an SCCS file  
**sccsfile(4)** .....format of SCCS file  
**scshdr(4)** .....section header for a common object file  
**script(1)** .....make typescript of terminal session  
**sdb(1)** .....symbolic debugger  
**sdiff(1)** .....side-by-side difference program  
**sed(1)** .....stream editor  
**seed48** (*See drand48(3C)*) .....generate uniformly distributed pseudo-random numbers  
**seekdir** (*See directory(3X)*) .....flexible length directory operation  
**select(2N)** .....synchronous I/O multiplexing  
**semctl(2)** .....semaphore control operations  
**semget(2)** .....get set of semaphores  
**semop(2)** .....semaphore operation  
**send(2N)** .....send a message from a socket

## COMMANDS

**sendmsg** (*See send(2N)*) ..... send a message from a socket  
**sendto** (*See send(2N)*) ..... send a message from a socket  
**services(4N)** ..... service name data base  
**setbuf(3S)** ..... assign buffering to a stream  
**setgid** (*See setuid(2)*) ..... set group ID  
**setgrent** (*See getgrent(3C)*) ..... obtain group file entry from a group file  
**sethostent** (*See gethostent(3N)*) ..... get network host entry  
**sethostid** (*See gethostid(2N)*) ..... set unique identifier of current host  
**sethostname** (*See gethostname(2N)*) ..... set name of current host  
**setjmp(3C)** ..... non-local goto  
**setkey** (*See crypt(3C)*) ..... generate DES encryption  
**setmnt(1M)** ..... establish mount table  
**setnetent** (*See getnetent(3N)*) ..... get network entry  
**setpggrp(2)** ..... set process group ID  
**setprotoent** (*See getprotoent(3N)*) ..... get protocol entry  
**setpwent** (*See getpwent(3C)*) ..... get password file entry  
**setregid(2)** ..... set real and effective group ID  
**setreuid(2)** ..... set real and effective user IDs  
**setservent** (*See getservent(3N)*) ..... get service entry  
**setsockopt** (*See getsockopt(2N)*) ..... set options on sockets  
**setuid(2)** ..... set user ID  
**setutent** (*See getut(3C)*) ..... access utmp file entry  
**setvbuf** (*See setbuf(3S)*) ..... assign buffering to a stream  
**setl** (*See spull(3X)*) ..... access long integer data in a machine independent fashion  
**sh(1)** ..... shell, the standard command programming language  
**sh(1)** ..... shell layer manager  
**shmctl(2)** ..... shared memory control operations  
**shmget(2)** ..... get shared memory segment  
**shmop(2)** ..... shared memory operations  
**shutacct** (*See acctsh(1M)*) ..... shell procedure for accounting  
**shutdown(1M)** ..... terminate all processing  
**shutdown(2N)** ..... shut down part of a full-duplex connection  
**sign(3F)** ..... Fortran transfer-of-sign intrinsic function  
**signal(2)** ..... specify what to do upon receipt of a signal  
**signal(3F)** ..... specify Fortran action on receipt of a system signal  
**sin** (*See trig(3M)*) ..... trigonometric function  
**sin(3F)** ..... Fortran sine intrinsic function  
**sinh(3F)** ..... Fortran hyperbolic sine intrinsic function  
**sinh(3M)** ..... hyperbolic function

## COMMANDS

**size(1)** .....print section sizes of common object files  
**size5.0(1)** .....size of an object file (System V a.out format only)  
**sleep(1)** .....suspend execution for an interval  
**sleep(3C)** .....suspend execution for interval  
**sngl (See ftype(3F))** .....explicit Fortran type conversion  
**sno(1)** .....SNOBOL interpreter  
**socket(2N)** .....create an endpoint for communication  
**sort(1)** .....sort and/or merge files  
**spell(1)** .....find spelling errors  
**spellin (See spell(1))** .....work with the spell program's hash lists  
**spline(1G)** .....interpolate smooth curve  
**split(1)** .....split a file into pieces  
**sprintf (See printf(3S))** .....print formatted output  
**swtbl(3X)** .....access long integer data in a machine independent fashion  
**sqrt (See exp(3M))** .....square root function  
**sqrt(3F)** .....Fortran square root intrinsic function  
**srand (See rand(3C))** .....simple random-number generator  
**srand (See rand(3F))** .....Fortran uniform random-number generator  
**srand48 (See drand48(3C))** .....generate uniformly distributed pseudo-random numbers  
**sscanf (See scanf(3S))** .....convert formatted input  
**ssignal(3C)** .....software signal  
**ssp(1)** .....make output single spaced  
**startup (See acctsh(1M))** .....shell procedure for accounting  
**stat(2)** .....get file status  
**stat(5)** .....data returned by stat system call  
**stdio(3S)** .....standard buffered input/output package  
**stdipc(3C)** .....standard interprocess communication package  
**stime(2)** .....set time  
**streat (See string(3C))** .....string operation  
**strchr (See string(3C))** .....string operation  
**stremap (See string(3C))** .....string operation  
**streqp(3F)** .....string comparison intrinsic function  
**strcpy (See string(3C))** .....string operation  
**strespn (See string(3C))** .....string operation  
**string(3C)** .....string operation  
**strings(1)** .....find the printable strings in an object, or other binary file  
**strip(1)** .....strip symbol and line number information from an object file  
**strip5.0(1)** .....remove symbols and relocation bits (System V a.out format only)  
**strlen (See string(3C))** .....string operation

## COMMANDS

**streqat** (*See string(3C)*) .....string operation  
**strncmp** (*See string(3C)*) .....string operation  
**strncpy** (*See string(3C)*) .....string operation  
**strpbrk** (*See string(3C)*) .....string operation  
**strrchr** (*See string(3C)*) .....string operation  
**strspn** (*See string(3C)*) .....string operation  
**strtod(3C)** .....convert string to double-precision number  
**strtok** (*See string(3C)*) .....string operation  
**strtol(3C)** .....convert string to integer  
**stty(1)** .....set the options for a terminal  
**su(1)** .....become super-user or another user  
**sum(1)** .....print checksum and block count of a file  
**sum7(1)** .....sum and count blocks in a file  
**sumdir(1)** .....sum and count characters in the files in the given directories  
**swab(3C)** .....swap bytes  
**sxt(7)** .....pseudo-device driver  
**syms(4)** .....common object file symbol table format  
**sync(1)** .....update the super block  
**sync(2)** .....update super-block  
**sys\_errlist** (*See perror(3C)*) .....system error message  
**sys\_nerr** (*See perror(3C)*) .....system error message  
**sysdef(1M)** .....system definition  
**system(3F)** .....issue a shell command from Fortran  
**system(3S)** .....issue a shell command  
**tabs(1)** .....set tabs on a terminal  
**tall(1)** .....deliver the last part of a file  
**take(1C)** .....takes a file from a remote machine  
**talk(1N)** .....talk to another user  
**tan** (*See trig(3M)*) .....trigonometric function  
**tan(3F)** .....Fortran tangent intrinsic function  
**tanh** (*See sinh(3M)*) .....hyperbolic function  
**tanh(3F)** .....Fortran hyperbolic tangent intrinsic function  
**tapesave** (*See filesave(1M)*) .....daily/weekly UNIX file system backup  
**tar(1)** .....tape file archiver  
**tbl(1)** .....format tables for nroff or troff  
**tc(1)** .....phototypesetter simulator  
**tcp(3P)** .....Internet Transmission Control Protocol  
**tdelete** (*See tsearch(3C)*) .....manage binary search trees  
**tee(1)** .....pipe fitting

## COMMANDS

**tellinit** (*See init(1M)*) ..... process control initialization

**telldir** (*See directory(3X)*) ..... flexible length directory operation

**telnet(1N)** ..... user interface to the TELNET protocol

**telnetd(8N)** ..... DARPA TELNET protocol server

**tempnam** (*See tmpnam(3S)*) ..... create a name for a temporary file

**term(4)** ..... format of compiled term file.

**term(5)** ..... conventional names for terminals

**termcap(3X)** ..... terminal independent operation routines

**termcap(5)** ..... terminal capability data base

**terminfo(4)** ..... terminal capability data base

**termio(7)** ..... general terminal interface

**test(1)** ..... condition evaluation command

**tfind** (*See tsearch(3C)*) ..... manage binary search trees

**tftpd(8N)** ..... DARPA Trivial File Transfer Protocol server

**tgetent** (*See termcap(3X)*) ..... terminal independent operation routine

**tgetflag** (*See termcap(3X)*) ..... terminal independent operation routine

**tgetnum** (*See termcap(3X)*) ..... terminal independent operation routine

**tgetstr** (*See termcap(3X)*) ..... terminal independent operation routine

**tgoto** (*See termcap(3X)*) ..... terminal independent operation routine

**tlc(1M)** ..... terminfo compiler

**time(1)** ..... time a command

**time(2)** ..... get time

**times(2)** ..... get process and child process times

**timex(1)** ..... time a command; report process data and system activity

**tmpfile(3S)** ..... create a temporary file

**tmpnam(3S)** ..... create a name for a temporary file

**toascii** (*See conv(3C)*) ..... translate characters

**tolower** (*See conv(3C)*) ..... translate characters

**touch(1)** ..... update access and modification times of a file

**toupper** (*See conv(3C)*) ..... translate characters

**tp(1)** ..... manipulate tape archive

**tplot(1G)** ..... graphics filters

**tput(1)** ..... query terminfo database

**tputs** (*See termcap(3X)*) ..... terminal independent operation routine

**tr(1)** ..... translate characters

**trek(6)** ..... trekkie game

**trig(3M)** ..... trigonometric functions

**troff(1)** ..... typeset text

**trpt(8N)** ..... transliterate protocol trace



## COMMANDS

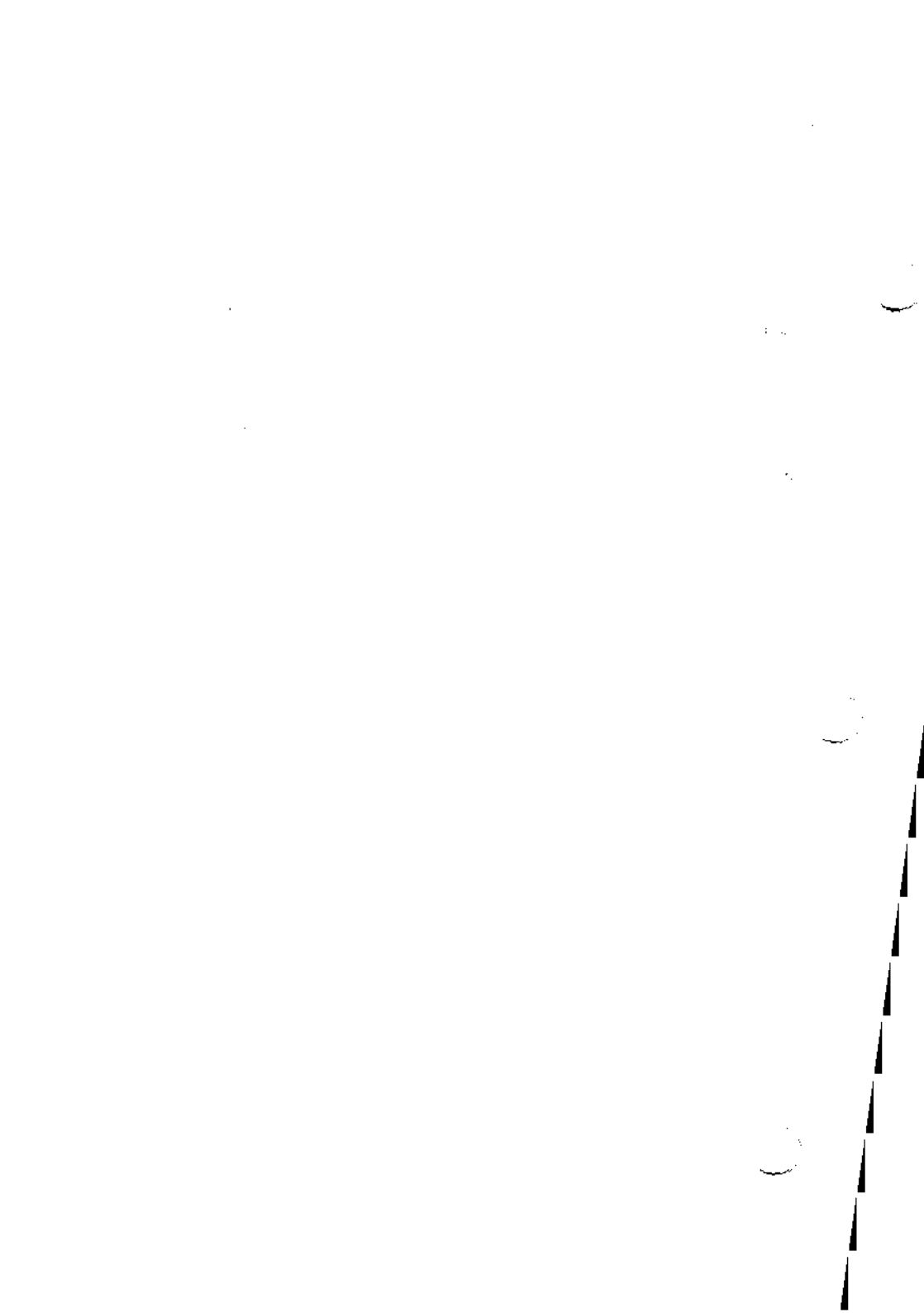
**true**(1) .....provide truth values  
**tsearch**(3C) .....manage binary search trees  
**tset**(1) .....set the teletype bits to a sensible state  
**tsort**(1) .....topological sort  
**ttt**(6) .....tic-tac-toe  
**tty**(1) .....get the terminal's name  
**tty**(7) .....controlling terminal interface  
**ttyname**(3C) .....find name of a terminal  
**ttyslot**(3C) .....find the slot in the utmp file of the current user  
**ttytype**(4) .....data base of terminal types by port  
**turnacct** (See *acctsh*(1M)) .....shell procedure for accounting  
**twalk** (See *tsearch*(3C)) .....manage binary search trees  
**twinkle**(6) .....twinkle stars on the screen  
**types**(5) .....primitive system data types  
**tzset** (See *time*(3C)) .....convert date and time to string  
**u3b** (See *machid*(1)) .....provide truth value about your processor type  
**u3b5** (See *machid*(1)) .....provide truth value about your processor type  
**udp**(SP) .....Internet User Datagram Protocol  
**ul**(1) .....do underlining  
**ulimit**(2) .....get and set user limits  
**umask**(1) .....set file-creation mode mask  
**umask**(2) .....set and get file creation mask  
**umount** (See *mount*(1M)) .....dismount file system  
**umount**(2) .....unmount a file system  
**uname**(1) .....print name of current UNIX system  
**uname**(2) .....get name of current UNIX system  
**unset**(1) .....undo a previous get of an SCCS file  
**unsetc**(3S) .....push character back into input stream  
**unq**(1) .....report repeated lines in a file  
**units**(1) .....conversion program  
**unlink** (See *link*(1M)) .....exercise unlink system call  
**unlink**(2) .....remove directory entry  
**unpack** (See *pack*(1)) .....expand compressed files  
**updater**(1) .....update files between two machines  
**updater**(1M) .....update files between two machines  
**ustat**(2) .....get file system statistics  
**utime**(2) .....set file access and modification times  
**utmp**(4) .....utmp entry format  
**utmpname** (See *getut*(3C)) .....access utmp file entry

## COMMANDS

<b>nuclean(1M)</b> .....	UUCP spool directory clean-up
<b>uucp(1C)</b> .....	UNIX system to UNIX system copy
<b>uulog</b> (See <i>uucp(1C)</i> ).....	prints a summary log of UUCP and UUX transactions
<b>uuname</b> (See <i>uucp(1C)</i> ).....	lists the UUCP names of known systems
<b>uupick</b> (See <i>uuto(1C)</i> ).....	public UNIX-to-UNIX system file copy
<b>uustat(1C)</b> .....	UUCP status inquiry and job control
<b>uusub(1M)</b> .....	monitor UUCP network
<b>uuto(1C)</b> .....	public UNIX-to-UNIX system file copy
<b>uux(1C)</b> .....	UNIX-to-UNIX system command execution
<b>uvar(2)</b> .....	returns system-specific configuration information
<b>val(1)</b> .....	validate SCCS file
<b>values(5)</b> .....	machine-dependent values
<b>varargs(5)</b> .....	handle variable argument list
<b>vax</b> (See <i>machid(1)</i> ).....	provide truth value about your processor type
<b>vc(1)</b> .....	version control
<b>vehk(1M)</b> .....	version checkup
<b>vedit</b> (See <i>vi(1)</i> ).....	screen-oriented (visual) display editor based on ex
<b>version(1)</b> .....	reports version number of files
<b>vfprintf</b> (See <i>vprintf(3S)</i> ).....	print formatted output of a varargs argument list
<b>vfprintf</b> (See <i>vprintf(3X)</i> ).....	print formatted output of a varargs argument list
<b>vi(1)</b> .....	screen-oriented (visual) display editor based on ex
<b>vview</b> (See <i>vi(1)</i> ).....	screen-oriented (visual) display editor based on ex
<b>vocabulary(1M)</b> .....	copy file systems with label checking
<b>vprintf(3S)</b> .....	print formatted output of a varargs argument list
<b>vprintf(3X)</b> .....	print formatted output of a varargs argument list
<b>vsprintf</b> (See <i>vprintf(3S)</i> ).....	print formatted output of a varargs argument list
<b>vsprintf</b> (See <i>vprintf(3X)</i> ).....	print formatted output of a varargs argument list
<b>wait(2)</b> .....	wait for child process to stop or terminate
<b>wait3(2N)</b> .....	wait for child process to stop or terminate
<b>wall(1M)</b> .....	write to all users
<b>wc(1)</b> .....	word count
<b>what(1)</b> .....	identify SCCS files
<b>whereis(1)</b> .....	locate source, binary, and/or manual for program
<b>who(1)</b> .....	who is on the system
<b>whoami(1)</b> .....	print effective current user id
<b>whode(1M)</b> .....	who is doing what
<b>worm(6)</b> .....	play the growing worm game
<b>worms(6)</b> .....	animate worms on a display terminal
<b>write(1)</b> .....	write to another user

## COMMANDS

**write(3)** .....write on a file  
**writer(3N)** .....write on a file  
**wtmp** (*See utmp(4)*) .....wtmp entry format  
**wtmpfix** (*See fwtmp(1M)*) .....manipulate connect accounting records  
**wump(6)** .....the game of hunt-the-wumpus  
**xargs(1)** .....construct argument list(s) and execute command  
**xor** (*See bool(3F)*) .....Fortran bitwise boolean function  
**xstr(1)** .....extract strings from C programs to implement shared strings  
**y0** (*See bessel(3M)*) .....Bessel function  
**y1** (*See bessel(3M)*) .....Bessel function  
**yacc(1)** .....yet another compiler-compiler  
**yn** (*See bessel(3M)*) .....Bessel function  
**zabs** (*See abs(3F)*) .....Fortran absolute value



## PERMUTED INDEX

functions of DASI 300 and/ 300, 300s: handle special . . . . . 300(1)  
 /special functions of DASI 300 and 300s terminals. . . . . 300(1)  
 of DASI 300 and 300s/ 300, 300s: handle special functions . . . . . 300(1)  
 functions of DASI 300 and 300s terminals. /special . . . . . 300(1)  
 I3tol, I0I3: convert between 3-byte integers and long/ I3tol(3C)  
 comparison. diff3: 3-way differential file . . . . . diff3(1)  
 Tektronix 4014 terminal. 4014: paginator for the . . . . . 4014(1)  
 paginator for the Tektronix 4014 terminal. 4014: . . . . . 4014(1)  
 of the DASI 450 terminal. 450: handle special functions . . . . . 450(1)  
 special functions of the DASI 450 terminal. 450: handle . . . . . 450(1)  
 f77: Fortran 77 compiler. . . . . f77(1)  
 long integer and base-64/ a64l, l64a: convert between . . . . . a64l(3C)  
 program. abort: generate an IOT fault. . . . . abort(3C)  
 Fortran absolute value. abort: terminate Fortran . . . . . abort(3F)  
 value. abs, iabs, dabs, cabs, zabs: . . . . . abs(3F)  
 abs: return integer absolute . . . . . abs(3C)  
 absolute value. . . . . abs(3C)  
 dabs, cabs, zabs: Fortran absolute value. abs, iabs, . . . . . abs(3F)  
 /floor, ceiling, remainder, absolute value functions. . . . . floor(3M)  
 socket. accept: accept a connection on a . . . . . accept(2N)  
 a socket. accept: accept a connection on . . . . . accept(2N)  
 LP requests. accept, reject: allow/prevent . . . . . accept(1M)  
 of a file. touch: update access and modification times . . . . . touch(1)  
 utime: set file access and modification times. . . . . utime(2)  
 accessibility of a file. access: determine . . . . . access(2)  
 machine/ sputl, sgetl: access long integer data in a . . . . . sputl(3X)  
 phys: allow a process to access physical addresses. . . . . phys(2)  
 ldfcn: common object file access routines. . . . . ldfcn(4)  
 copy file systems for optimal access time. dcopy: . . . . . dcopy(1M)  
 /setuent, enduent, utmpname: access utmp file entry. . . . . getut(3C)  
 access: determine accessibility of a file. . . . . access(2)  
 enable or disable process accounting. acct: . . . . . acct(2)  
 acctcon2: connect-time accounting. acctcon1, . . . . . acctcon(1M)  
 acctprcl, acctprc2: process accounting. . . . . acctprc(1M)  
 turnacct: shell procedures for accounting. /startup, . . . . . acctsh(1M)  
 /accton, acctwtmp: overview of accounting and miscellaneous/ . . . . . acct(1M)  
 accounting and miscellaneous accounting commands. /of . . . . . acct(1M)  
 diskug: generate disk accounting data by user.ID. . . . . diskug(1M)  
 acct: per-process accounting file format. . . . . acct(4)  
 search and print process accounting file(s). acctcom: . . . . . acctcom(1)  
 acctmerg: merge or add total accounting files. . . . . acctmerg(1M)  
 mlock: return Fortran time accounting. . . . . mlock(3F)  
 summary from per-process accounting records. /command . . . . . acctcms(1M)  
 wtmpfix: manipulate connect accounting records. fwtmp, . . . . . fwtmp(1M)  
 runacct: run daily accounting. . . . . runacct(1M)  
 process accounting. acct: enable or disable . . . . . acct(2)  
 file format. acct: per-process accounting . . . . . acct(4)  
 per-process accounting/ acctcms: command summary from . . . . . acctcms(1M)  
 process accounting file(s). acctcom: search and print . . . . . acctcom(1)  
 connect-time accounting. acctcon1, acctcon2: . . . . . acctcon(1M)  
 accounting. acctcon1, acctcon2: connect-time . . . . . acctcon(1M)  
 acctwtmp: overview of acctdisk, acctdusg, accton, . . . . . acct(1M)  
 overview of/ acctdisk, acctdusg, accton, acctwtmp: . . . . . acct(1M)  
 accounting files. acctmerg: merge or add total . . . . . acctmerg(1M)  
 acctdisk, acctdusg, accton, acctwtmp: overview of/ . . . . . acct(1M)  
 accounting. acctprcl, acctprc2: process . . . . . acctprc(1M)  
 acctprc2: process accounting. acctprc: overview of/ . . . . . acctprc(1M)  
 acctdisk, acctdusg, accton, acctwtmp: overview of/ . . . . . acct(1M)  
 sin, cos, tan, asin, trig(3M)  
 intrinsic function. acos, atan, atan2:/ . . . . . acos(3F)  
 killall: kill all acos, dacos: Fortran arccosine . . . . . acos(3F)  
 active processes. . . . . killall(1M)

sag: system  
 sa1, sa2, sacc: system  
 sar: system  
 current SCCS file editing  
 report process data and system  
 random, hopefully interesting,  
 formatting/ modsd: the OSDD  
 acctmrg: merge or  
 putenv: change or  
 /inet\_netof: Internet  
 arp:  
 a process to access physical  
 SCCS files.  
 admin: create and  
 game.  
 imaginary part of complex/  
 part intrinsic function.  
 alarm: set a process's  
 clock.  
 delivermail.  
 aliases:  
 earth. aliens: The  
 attack the earth.  
 Change data segment space  
 realloc, calloc: main memory  
 mallinfo: fast main memory  
 physical addresses. phys:  
 accept, reject:  
 natural logarithm/ log,  
 logarithm intrinsic/ log10,  
 information for bad block/  
 for bad block/ altblk:  
 Fortran/ max, max0,  
 max, max0, amax0, max1,  
 Fortran/ min, min0,  
 min, min0, amin0, min1,  
 remaindering intrinsic/ mod,  
 rshift: Fortran bitwise/  
 /locate source, binary,  
 sort: sort  
 terminal. worms:  
 rain:  
 Fortran nearest integer/  
 bcd: convert to  
 link editor output.  
 files. aouthdr.h -  
 editor output.  
 common object files.  
 introduction to commands and  
 maintenance commands and  
 maintainer for portable/  
 format.  
 maintainer.  
 format.  
 number: convert  
 delivermail: deliver mail to  
 language. bc:  
 acos, dacos: Fortran  
 maintainer. ar5.0:  
 for portable archives. ar:  
 cpio: format of cpio  
 ar: common  
 header of a member of an  
 an archive/ ldahread: read the  
 activity graph. . . . .  
 activity report package. . . . .  
 activity reporter. . . . .  
 activity. sact: print . . . . .  
 activity. /time a command; . . . . .  
 adage. fortune: print a . . . . .  
 adapter macro package for . . . . .  
 add total accounting files. . . . .  
 add value to environment. . . . .  
 address manipulation routines. . . . .  
 Address Resolution Protocol. . . . .  
 addresses. phys: allow . . . . .  
 admin: create and administer . . . . .  
 administer SCCS files. . . . .  
 adventure: an exploration . . . . .  
 aimag, dimag: Fortran . . . . .  
 aint, dint: Fortran integer . . . . .  
 alarm clock. . . . .  
 alarm: set a process's alarm . . . . .  
 aliases: aliases file for . . . . .  
 aliases file for delivermail. . . . .  
 alien invaders attack the . . . . .  
 aliens: The alien invaders . . . . .  
 allocation. brk, sbrk: . . . . .  
 allocator. malloc, free, . . . . .  
 allocator. /calloc, mallopt, . . . . .  
 allow a process to access . . . . .  
 allow/prevent LP requests. . . . .  
 alog, dlog, clog: Fortran . . . . .  
 alog10, dlog10: Fortran common . . . . .  
 altblk: alternate block . . . . .  
 alternate block information . . . . .  
 amax0, max1, amax1, dmax1: . . . . .  
 amax1, dmax1: Fortran/ . . . . .  
 amin0, min1, amin1, dmin1: . . . . .  
 amin1, dmin1: Fortran/ . . . . .  
 amod, dmod: Fortran . . . . .  
 and, or, xor, not, lshift, . . . . .  
 and/or manual for program. . . . .  
 and/or merge files. . . . .  
 animate worms on a display . . . . .  
 animated raindrops display. . . . .  
 anint, dnint, nint, idnint: . . . . .  
 antique media. . . . .  
 a.out: common assembler and . . . . .  
 a.out header for common object . . . . .  
 a.out5.0: assembler and link . . . . .  
 aouthdr.h - a.out header for . . . . .  
 application programs. intro: . . . . .  
 application programs. /system . . . . .  
 ar: archive and library . . . . .  
 ar: common archive file . . . . .  
 ar5.0: archive and library . . . . .  
 ar5.0: archive (library) file . . . . .  
 Arabic numerals to English. . . . .  
 arbitrary people. . . . .  
 arbitrary-precision arithmetic . . . . .  
 arccosine intrinsic function. . . . .  
 archive and library . . . . .  
 archive and library maintainer . . . . .  
 archive. . . . .  
 archive file format. . . . .  
 archive file. /the archive . . . . .  
 archive header of a member of . . . . .  
 sag(1G)  
 sar(1M)  
 sar(1)  
 sact(1)  
 timex(1)  
 fortune(6)  
 modsd(5)  
 acctmrg(1M)  
 putenv(3C)  
 inet(3N)  
 arp(5P)  
 phys(2)  
 admin(1)  
 admin(1)  
 adventure(6)  
 aimag(3F)  
 aint(3F)  
 alarm(2)  
 alarm(2)  
 aliases(7N)  
 aliases(7N)  
 aliens(6)  
 aliens(6)  
 brk(2)  
 malloc(3C)  
 malloc(3X)  
 phys(2)  
 accept(1M)  
 log(3F)  
 log10(3F)  
 altblk(4)  
 altblk(4)  
 max(3F)  
 max(3F)  
 min(3F)  
 min(3F)  
 mod(3F)  
 bool(3F)  
 whereis(1)  
 sort(1)  
 worms(6)  
 rain(6)  
 round(3F)  
 bcd(6)  
 a.out(4)  
 aouthdr(4)  
 a.out5.0(4)  
 aouthdr(4)  
 intro(1)  
 intro(1M)  
 ar(1)  
 ar(4)  
 ar5.0(1)  
 ar5.0(4)  
 number(6)  
 delivermail(8N)  
 bc(1)  
 acos(3F)  
 ar5.0(1)  
 ar(1)  
 cpio(4)  
 ar(4)  
 ldahread(3X)  
 ldahread(3X)

ar5.0:	archive (library) file format. . . . .	ar5.0(4)
tp: manipulate tape	archive. . . . .	tp(1)
tar: tape file	archiver. . . . .	tar(1)
maintainer for portable	archives. /archive and library . . . . .	ar(1)
cpio: copy file	archives in and out. . . . .	cpio(1)
asin, dasin: Fortran	arcsine intrinsic function. . . . .	asin(3F)
atan2, datan2: Fortran	arctangent intrinsic function. . . . .	atan2(3F)
atan, datan: Fortran	arctangent intrinsic function. . . . .	atan(3F)
imaginary part of complex	argument. /dimag: Fortran . . . . .	aimag(3F)
return Fortran command-line	argument. getarg: . . . . .	getarg(3F)
varargs: handle variable	argument list. . . . .	varargs(5)
formatted output of a varargs	argument list. /print . . . . .	vprintf(3S)
formatted output of a varargs	argument list. /print . . . . .	vprintf(3X)
command. xargs: construct	argument list(s) and execute . . . . .	xargs(1)
getopt: get option letter from	argument vector. . . . .	getopt(3C)
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)
Protocol.	arp: Address Resolution . . . . .	arp(5P)
expr: evaluate arguments	as an expression. . . . .	expr(1)
	as: common assembler. . . . .	as(1)
characters. asa: interpret	as5.0: assembler. . . . .	as5.0(1)
control characters.	ASA carriage control . . . . .	asa(1)
ascii: map of	asa: interpret ASA carriage . . . . .	asa(1)
set.	ASCII character set. . . . .	ascii(5)
long integer and base-64	ascii: map of ASCII character . . . . .	ascii(5)
number. atof: convert	ASCII string. /convert between . . . . .	a64l(3C)
and/ ctime, localtime, gmtime,	ASCII string to floating-point . . . . .	atof(3C)
trigonometric/ sin, cos, tan,	asctime, tzset: convert date . . . . .	ctime(3C)
intrinsic function.	asin, acos, atan, atan2: . . . . .	trig(3M)
help:	asin, dasin: Fortran arcsine . . . . .	asin(3F)
output. a.out: common	ask for help in using SCCS. . . . .	help(1)
output. a.out5.0:	assembler and link editor . . . . .	a.out(4)
as: common	assembler and link editor . . . . .	a.out5.0(4)
as5.0:	assembler. . . . .	as(1)
assertion.	assembler. . . . .	as5.0(1)
assert: verify program	assert: verify program . . . . .	assert(3X)
setbuf, setvbuf:	assertion. . . . .	assert(3X)
a later time.	assign buffering to a stream. . . . .	setbuf(3S)
sin, cos, tan, asin, acos,	at, batch: execute commands at . . . . .	at(1)
arctangent intrinsic/	atan, atan2: trigonometric/ . . . . .	trig(3M)
arctangent intrinsic/	atan, datan: Fortran . . . . .	atan(3F)
cos, tan, asin, acos, atan,	atan2, datan2: Fortran . . . . .	atan2(3F)
floating-point number.	atan2: trigonometric/ sin, . . . . .	trig(3M)
double-precision/ strtod,	atof: convert ASCII string to . . . . .	atof(3C)
integer. strtol, atol,	atof: convert string to . . . . .	strtod(3C)
integer. strtol,	atoi: convert string to . . . . .	strtol(3C)
aliens: The alien invaders	atol, atoi: convert string to . . . . .	strtol(3C)
autorobots: Escape from the	attack the earth. . . . .	aliens(6)
automatic robots.	automatic robots. . . . .	autorobots(6)
lav: print load	autorobots: Escape from the . . . . .	autorobots(6)
processing language.	average statistics. . . . .	lav(1)
ungetc: push character	awk: pattern scanning and . . . . .	awk(1)
	back into input stream. . . . .	ungetc(3S)
	back: the game of backgammon. . . . .	back(6)
	backgammon. . . . .	back(6)
back: the game of	backup. filesave, tapesave: . . . . .	filesave(1M)
daily/weekly UNIX file system	backup. . . . .	finc(1M)
finc: fast incremental	backup tape. . . . .	frec(1M)
frec: recover files from a	bad block handling. /alternate . . . . .	altblk(4)
block information for	bad block information. . . . .	badblk(1M)
/program to set or update	badblk: program to set or . . . . .	badblk(1M)
update bad block information.	banner: make posters. . . . .	banner(1)

banner7: print large printer.	banner on printer.	banner7(1)
hosts: host name data	banner7: print large banner on	banner7(1)
networks: network name data	base.	hosts(4N)
port. ttytype: data	base.	networks(4N)
protocols: protocol name data	base of terminal types by	ttytype(4)
services: service name data	base.	protocols(4N)
terminal capability data	base.	services(4N)
terminal capability data	base. termcap:	termcap(5)
between long integer and (visual) display editor	base. terminfo:	terminfo(4)
portions of path names.	base-64 ASCII string. /convert	a64i(3C)
later time. at, arithmetic language.	based on ex. /screen-oriented	vi(1)
system initialization/ brc, string operations. bcopy, and byte string operations.	basename, dirname: deliver	basename(1)
	batch: execute commands at a	at(1)
	bc: arbitrary-precision	bc(1)
	bcd: convert to antique media.	bcd(6)
	bcheckrc, rc, powerfail:	brc(1M)
	bcmp, bzero, ffs: bit and byte	bstring(3N)
	bcopy, bcmp, bzero, ffs: bit	bstring(3N)
	bcopy: interactive block copy.	bcopy(1M)
	bdiff: big diff.	bdiff(1)
	beautifier.	cb(1)
	Bessel functions.	bessel(3M)
	bfs: big file scanner.	bfs(1)
	binary, and/or manual for/	whereis(1)
	binary directories.	cpset(1M)
	binary file. /the printable	strings(1)
	binary input/output.	fread(3S)
	binary search a sorted table.	bsearch(3C)
	binary search trees. tsearch,	tsearch(3C)
	bind: bind a name to a socket.	bind(2N)
	bind: bind a name to a socket.	bind(2N)
	bit and byte string/	bstring(3N)
	bits. strip5.0:	strip5.0(1)
	bits to a sensible state.	tset(1)
	bitwise boolean functions.	bool(3F)
	bj: the game of black jack.	bj(6)
	black jack.	bj(6)
	block copy.	bcopy(1M)
	block count of a file.	sum(1)
	block handling. /alternate	altblk(4)
	block information. badblk:	badblk(1M)
	block information for bad	altblk(4)
	block.	sync(1)
	block transfer data.	blt(3C)
	blocks.	df(1M)
	blocks in a file.	sum7(1)
	blt, blt512: block transfer	blt(3C)
	blt512: block transfer data.	blt(3C)
	B-NET.	netmailer(8N)
	B-NET network mail system.	netmail(8N)
	boolean functions. /lshift,	bool(3F)
	boot: startup procedures.	boot(8)
	brc, bcheckrc, rc, powerfail:	brc(1M)
	brk, sbrk: change data segment	brk(2)
	bs: a compiler/interpreter for	bs(1)
	bsearch: binary search a	bsearch(3C)
	buffered input/output package.	stdio(3S)
	buffering to a stream.	setbuf(3S)
	build special file.	mknod(1M)
	byte order. /convert values	byteorder(3N)
	byte string operations.	bstring(3N)
	bytes.	swab(3C)
	bzero, ffs: bit and byte	bstring(3N)
	cc: C compiler.	cc(1)
cb: C program		
j0, j1, jn, y0, y1, yn:		
whereis: locate source,		
cpset: install object files in		
strings in an object, or other		
fread, fwrite:		
bsearch:		
tfind, tdelete, twalk: manage		
bind:		
bcopy, bcmp, bzero, ffs:		
remove symbols and relocation		
/set or reset the teletype		
/not, lshift, rshift: Fortran		
bj: the game of		
bcopy: interactive		
sum: print checksum and		
block information for bad		
program to set or update bad		
block/ altblk: alternate		
sync: update the super		
blt, blt512:		
df: report number of free disk		
sum7: sum and count		
data.		
blt,		
netmailer: deliver mail to		
netmail: the		
rshift: Fortran bitwise		
system initialization shell/		
space allocation.		
modest-sized programs.		
sorted table.		
stdio: standard		
setbuf, setvbuf: assign		
mknod:		
between host and network		
/bcmp, bzero, ffs: bit and		
swab: swap		
string/ bcopy, bcmp,		
cc:		



cc5.0:	C compiler. . . . .	cc5.0(1)
mc68cc:	C compiler. . . . .	mc68cc(1)
cflow: generate	C flowgraph. . . . .	cflow(1)
cpp: the	C language preprocessor. . . . .	cpp(1)
cpp: the	C language preprocessor. . . . .	cpp5.0(1)
cb:	C program beautifier. . . . .	cb(1)
lint: a	C program checker. . . . .	lint(1)
cxref: generate	C program cross-reference. . . . .	cxref(1)
maintain a tags file for a	C program. ctags: . . . . .	ctags(1)
ctrace:	C program debugger. . . . .	ctrace(1)
xstr: extract strings from	C programs to implement shared/ . . . . .	xstr(1)
message file by massaging	C source. /create an error . . . . .	mkstr(1)
value. abs, iabs, dabs,	cabs, zabs: Fortran absolute . . . . .	abs(3F)
	cal: print calendar. . . . .	cal(1)
	calculator. . . . .	dc(1)
dc: desk	calendar. . . . .	cal(1)
cal: print	calendar: reminder service. . . . .	calendar(1)
	call another UNIX system. . . . .	cu(1C)
	cu:	stat(5)
data returned by stat system	call. stat: . . . . .	stat(5)
malloc, free, realloc,	calloc: main memory allocator. . . . .	malloc(3C)
fast/ malloc, free, realloc,	calloc, mallopt, mallinfo: . . . . .	malloc(3X)
intro: introduction to system	calls and error numbers. . . . .	intro(2)
link and unlink system	calls. link, unlink: exercise . . . . .	link(1M)
to an LP line printer. lp,	cancel: send/cancel requests . . . . .	lp(1)
termcap: terminal	capability data base. . . . .	termcap(5)
terminfo: terminal	capability data base. . . . .	terminfo(4)
cribbage: the	card game cribbage. . . . .	cribbage(6)
pnch: file format for	card images. . . . .	pnch(4)
asa: interpret ASA	carriage control characters. . . . .	asa(1)
files.	cat: concatenate and print . . . . .	cat(1)
	cb: C program beautifier. . . . .	cb(1)
	cc: C compiler. . . . .	cc(1)
	cc5.0: C compiler. . . . .	cc5.0(1)
function. cos, dcos,	ccos: Fortran cosine intrinsic . . . . .	cos(3F)
	cd: change working directory. . . . .	cd(1)
commentary of an SCCS delta.	cdc: change the delta . . . . .	cdc(1)
ceiling, remainder,/ floor,	ceil, fmod, fabs: floor, . . . . .	floor(3M)
/ceil, fmod, fabs: floor,	ceiling, remainder, absolute/ . . . . .	floor(3M)
intrinsic/ exp, dexp,	cexp: Fortran exponential . . . . .	exp(3F)
	cflow: generate C flowgraph. . . . .	cflow(1)
	(change) to an SCCS file. . . . .	delta(1)
delta: make a delta	channel. . . . .	pipe(2)
pipe: create an interprocess	char: explicit Fortran type/ . . . . .	ftype(3F)
/dble, cmplx, dcmplx, ichar,	character back into input . . . . .	ungetc(3S)
stream. ungetc: push	character definitions for eqn . . . . .	eqnchar(5)
and neqn. eqnchar: special	character frequencies in a . . . . .	freq(1)
file. freq: report on	character login name of the . . . . .	cuserid(3S)
user. cuserid: get	character or word from a/ . . . . .	getc(3S)
/getchar, fgetc, getw: get	character or word on a stream. . . . .	putc(3S)
/putchar, fputc, putw: put	character set. . . . .	ascii(5)
ascii: map of ASCII	characters. asa: . . . . .	asa(1)
interpret ASA carriage control	characters. /_toupper, . . . . .	conv(3C)
_tolower, toascii: translate	characters. /isprint, isgraph, . . . . .	ctype(3C)
isctrl, isascii: classify	characters in the files in the . . . . .	sumdir(1)
given/ sumdir: sum and count	characters. . . . .	tr(1)
tr: translate	chargefee, ckpact, dodisk, . . . . .	acctsh(1M)
lastlogin, monacct, nulladm,/	chase: Try to escape the . . . . .	chase(6)
killer robots.	chdir: change working . . . . .	chdir(2)
directory.	check and interactive repair. . . . .	fscck(1M)
/dfscck: file system consistency	checkall: faster file system . . . . .	checkall(1M)
checking procedure.	checkcw: prepare . . . . .	cw(1)
constant-width text for/ cw,	checkeq: format mathematical . . . . .	eqn(1)
text for nroff or/ eqn, neqn,	checker. . . . .	lint(1)
lint: a C program	checkers. pwck, . . . . .	pwck(1M)
grpck: password/group file		

checkall: faster file system	checking procedure. . . . .	checkall(1M)
copy file systems with label	checking. volcopy, labelit: . . . . .	volcopy(1M)
systems processed by fsck.	checklist: list of file . . . . .	checklist(4)
formatted with the/ mm, osdd,	checkmtn: print/check documents . . . . .	mm(1)
file. sum: print	checksum and block count of a . . . . .	sum(1)
vchk: version	checkup. . . . .	vchk(1M)
system nodename.	chgnod: change current UNIX . . . . .	chgnod(1M)
chown,	chgrp: change owner or group. . . . .	chown(1)
times: get process and	child process times. . . . .	times(2)
terminate. wait: wait for	child process to stop or . . . . .	wait(2)
terminate. wait3: wait for	child process to stop or . . . . .	wait3(2N)
	chmod: change mode. . . . .	chmod(1)
	chmod: change mode of file. . . . .	chmod(2)
of a file.	chown: change owner and group . . . . .	chown(2)
group.	chown, chgrp: change owner or . . . . .	chown(1)
	chroot: change root directory. . . . .	chroot(2)
for a command.	chroot: change root directory . . . . .	chroot(1M)
monacct, nulladm,/ chargefee,	ckpact, dodisk, lastlogin, . . . . .	acctsh(1M)
isgraph, iscntrl, isascii:	classify characters. /isprint, . . . . .	ctype(3C)
uuclean: uucp spool directory	clean-up. . . . .	uuclean(1M)
	clear: clear terminal screen. . . . .	clear(1)
	clri: clear i-node. . . . .	clri(1M)
	clear: clear terminal screen. . . . .	clear(1)
status/ ferror, feof,	clearerr, fileno: stream . . . . .	ferror(3S)
(command interpreter) with	C-like syntax. csh: a shell . . . . .	csh(1)
alarm: set a process's alarm	clock. . . . .	alarm(2)
cron:	clock daemon. . . . .	cron(1M)
	clock: report CPU time used. . . . .	clock(3C)
logarithm/ log, alog, dlog,	clog: Fortran natural . . . . .	log(3F)
ldclose, ldaclose:	close a common object file. . . . .	ldclose(3X)
close:	close a file descriptor. . . . .	close(2)
descriptor.	close: close a file . . . . .	close(2)
fclose, fflush:	close or flush a stream. . . . .	fclose(3S)
/telldir, seekdir, rewinddir,	closedir: flexible length/ . . . . .	directory(3X)
	clri: clear i-node. . . . .	clri(1M)
	cmp: compare two files. . . . .	cmp(1)
/real, float, singl, dble,	cmpix, dcmpix, ichar, char:/ . . . . .	ftype(3F)
line-feeds.	col: filter reverse . . . . .	col(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)
nice: run a	command at low priority. . . . .	nice(1)
change root directory for a	command. chroot: . . . . .	chroot(1M)
env: set environment for	command execution. . . . .	env(1)
uux: UNIX-to-UNIX system	command execution. . . . .	uux(1C)
system: issue a shell	command from Fortran. . . . .	system(3F)
only). nohup: run a	command immune to hangups (sh . . . . .	nohup(1)
C-like syntax. csh: a shell	(command interpreter) with . . . . .	csh(1)
getopt: parse	command options. . . . .	getopt(1)
/shell, the standard/restricted	command programming language. . . . .	sh(1)
returning a stream to a remote	command. /routines for . . . . .	rcmd(3N)
and system/ timex: time a	command; report process data . . . . .	timex(1)
return stream to a remote	command. rexec: . . . . .	rexec(3N)
per-process/ acctcms:	command summary from . . . . .	acctcms(1M)
system: issue a shell	command. . . . .	system(3S)
test: condition evaluation	command. . . . .	test(1)
time: time a	command. . . . .	time(1)
argument list(s) and execute	command. xargs: construct . . . . .	xargs(1)
getarg: return Fortran	command-line argument. . . . .	getarg(3F)
and miscellaneous accounting	commands. /of accounting . . . . .	acct(1M)
intro: introduction to	commands and application/ . . . . .	intro(1)
/to system maintenance	commands and application/ . . . . .	intro(1M)
at, batch: execute	commands at a later time. . . . .	at(1)
install: install	commands. . . . .	install(1M)

cdc: change the delta	commentary of an SCCS delta.	cdc(1)
ar:	common archive file format.	ar(4)
editor output. a.out:	common assembler and link	a.out(4)
as:	common assembler.	as(1)
log10, alog10, dlog10: Fortran	common logarithm intrinsic/	log10(3F)
routines. ldfcn:	common object file access	ldfcn(4)
ldopen, ldaopen: open a	common object file for/	ldopen(3X)
/line number entries of a	common object file function.	ldhread(3X)
ldclose, ldaclose: close a	common object file.	ldclose(3X)
read the file header of a	common object file. ldhread:	ldhread(3X)
entries of a section of a	common object file. /number	ldlseek(3X)
the optional file header of a	common object file. /seek to	ldohseek(3X)
/entries of a section of a	common object file.	ldrseek(3X)
/section header of a	common object file.	ldhread(3X)
an indexed/named section of a	common object file. /seek to	ldsseek(3X)
of a symbol table entry of a	common object file. /the index	ldtbindex(3X)
symbol table entry of a	common object file. /indexed	ldtbread(3X)
seek to the symbol table of a	common object file. ldtbseek:	ldtbseek(3X)
line number entries in a	common object file. linenum:	linenum(4)
nm: print name list of	common object file.	nm(1)
relocation information for a	common object file. reloc:	reloc(4)
scnhdr: section header for a	common object file.	scnhdr(4)
table format. syms:	common object file symbol	syms(4)
aouthdr.h - a.out header for	common object files.	aouthdr(4)
filehdr: file header for	common object files.	filehdr(4)
ld: link editor for	common object files.	ld(1)
size: print section sizes of	common object files.	size(1)
comm: select or reject lines	common to two sorted files.	comm(1)
ipcs: report inter-process	communication facilities/	ipcs(1)
ftok: standard interprocess	communication package.	stidpc(3C)
socket: create an endpoint for	communication.	socket(2N)
diff: differential file	comparator.	diff(1)
cmp:	compare two files.	cmp(1)
SCCS file. scsdiff:	compare two versions of an	scsdiff(1)
lge, lgt, lle, lit: string	comparison intrinsic/	strcmp(3F)
diff3: 3-way differential file	comparison.	diff3(1)
dircmp: directory	comparison.	dircmp(1)
expression. regcmp, regex:	compile and execute a regular	regcmp(3X)
regex: regular expression	compile and match routines.	regex(5)
regcmp: regular expression	compile.	regcmp(1)
term: format of	compiled term file..	term(4)
cc: C	compiler.	cc(1)
cc5.0: C	compiler.	cc5.0(1)
f77: Fortran 77	compiler.	f77(1)
mc68cc: C	compiler.	mc68cc(1)
tic: terminfo	compiler.	tic(1M)
yacc: yet another	compiler-compiler.	yacc(1)
modest-sized programs. bs: a	compiler/interpreter for	bs(1)
erf, erf: error function and	complementary error function.	erf(3M)
Fortran imaginary part of	complex argument. /dimag:	aimag(3F)
conj, dconj: Fortran	complex conjugate intrinsic/	conj(3F)
pack, pcat, unpack:	compress and expand files.	pack(1)
table entry of a/ ldtbindex:	compute the index of a symbol	ldtbindex(3X)
cat:	concatenate and print files.	cat(1)
test:	condition evaluation command.	test(1)
uvar: returns system-specific	config: configure system.	config(1M)
parameters. ifconfig:	configuration information.	uvar(2)
config:	configure network interface	ifconfig(8N)
system. lpadmin:	configure system.	config(1M)
conjugate intrinsic function.	configure the LP spooling	lpadmin(1M)
conj, dconj: Fortran complex	conj, dconj: Fortran complex	conj(3F)
conjugate intrinsic function.	conjugate intrinsic function.	conj(3F)
fwtmp, wtmpfx: manipulate	connect accounting records.	fwtmp(1M)
on a socket.	connect: initiate a connection	connect(2N)

getpeername: get name of an out-going terminal line	connected peer. . . . .	getpeername(2N)
accept: accept a connection on a socket.	connection. dial: establish . . . . .	dial(3C)
down part of a full-duplex listen: listen for acctcon1, acctcon2:	connection on a socket. . . . .	accept(2N)
fsck, dfscck: file system connect-time accounting.	connection. shutdown: shut connections on a socket. . . . .	connect(2N)
math: math functions and cw, checkcw: prepare mkfslb:	connections on a socket. . . . .	shutdown(2N)
mkfs: construct a file system.	connect-time accounting. . . . .	listen(2N)
execute command. xargs: nroff/troff, tbl, and eqn ls: list csplit:	consistency check and/ constants. . . . .	acctcon(1M)
asa: interpret ASA carriage ioctl: fcntl: file init, telinit: process msgctl: message semctl: semaphore shmctl: shared memory fcntl: file tcp: Internet Transmission uucp status inquiry and job vc: version interface. tty:	constant-width text for troff. construct a file system. . . . .	fsck(1M)
terminals. term: char: explicit Fortran type units: dd: English, number: floating-point number. atof: integers and/ l3tol, ltol3: and base-64 ASCII/ a64l, l64a: /gmtime, asctime, tzset: and VAX-11/780/ fscv: to string. ecvt, fcvt, gcvt: scanf, fscanf, sscanf: strtod, atof: strtol, atol, atoi: bcd: htonl, hton, ntohl, ntohs: conv: object file dd: convert and bcopy: interactive block cpio: access time. dcopy: checking. volcopy, labelit: cp, ln, mv: rcp: remote file UNIX system to UNIX system UNIX-to-UNIX system file core: format of mem, kmem: cosine intrinsic function. atan2: trigonometric/ sin, hyperbolic cosine intrinsic/ functions. sinh, cos, dcosh, ccosh: Fortran /dcosh: Fortran hyperbolic	control. . . . .	math(5)
	control initialization. . . . .	kw(1)
	control operations. . . . .	mkfslb(1M)
	control operations. . . . .	mkfs(1M)
	control operations. . . . .	xargs(1)
	control options. . . . .	deroff(1)
	Control Protocol. . . . .	ls(1)
	control. uustat: . . . . .	csplit(1)
	control. . . . .	asa(1)
	controlling terminal conv: object file converter. . . . .	ioctl(2)
	conventional names for conversion. /dcmplx, ichar, conversion program. . . . .	fcntl(2)
	convert and copy a file. . . . .	init(1M)
	convert Arabic numerals to convert ASCII string to convert between 3-byte convert between long integer convert date and time to/ convert files between M68000 convert floating-point number convert formatted input. . . . .	msgctl(2)
	convert string to/ convert string to integer. . . . .	semctl(2)
	convert to antique media. . . . .	shmctl(2)
	convert values between host/ converter. . . . .	fcntl(5)
	copy a file. . . . .	tcp(5P)
	copy. . . . .	uustat(1C)
	copy file archives in and out. . . . .	vc(1)
	copy file systems for optimal copy file systems with label copy, link or move files. . . . .	tty(7)
	copy. . . . .	conv(1)
	copy. uucp, uulog, uname: . . . . .	term(5)
	copy. uuto, uupick: public core: format of core image core image file. . . . .	ftype(3F)
	core memory. . . . .	units(1)
	cos, dcosh, ccosh: Fortran cos, tan, asin, acos, atan, cosh, dcosh: Fortran cosh, tanh: hyperbolic cosine intrinsic function. . . . .	dd(1)
	cosine intrinsic function. . . . .	number(6)
	cos. cos, ccosh: Fortran cosine intrinsic function. . . . .	atof(3C)
		l3tol(3C)
		a64l(3C)
		ctime(3C)
		fscv(1M)
		ecvt(3C)
		scanf(3S)
		strtod(3C)
		strtol(3C)
		bcd(6)
		byteorder(3N)
		conv(1)
		dd(1)
		bcopy(1M)
		cpio(1)
		dcopy(1M)
		volcopy(1M)
		cp(1)
		rep(1N)
		uucp(1C)
		uuto(1C)
		core(4)
		core(4)
		mem(7)
		cos(3F)
		trig(3M)
		cosh(3F)
		sinh(3M)
		cos(3F)
		cosh(3F)

sum7: sum and	count blocks in a file. . . . .	sum7(1)
in the given/ sumdir: sum and	count characters in the files . . . . .	sumdir(1)
sum: print checksum and block	count of a file. . . . .	sum(1)
wc: word	count. . . . .	wc(1)
files.	cp, ln, mv: copy, link or move . . . . .	cp(1)
cpio: format of	cpio archive. . . . .	cpio(4)
and out.	cpio: copy file archives in . . . . .	cpio(1)
	cpio: format of cpio archive. . . . .	cpio(4)
preprocessor.	cpp: the C language . . . . .	cpp(1)
preprocessor.	cpp: the C language . . . . .	cpp5.0(1)
binary directories.	cpset: install object files in . . . . .	cpset(1M)
clock: report	CPU time used. . . . .	clock(3C)
craps: the game of	craps. . . . .	craps(6)
	craps: the game of craps. . . . .	craps(6)
system crashes.	crash: what to do when the . . . . .	crash(8)
what to do when the system	crashes. crash: . . . . .	crash(8)
rewrite an existing one.	creat: create a new file or . . . . .	creat(2)
file. tmpnam, tmpnam:	create a name for a temporary . . . . .	tmpnam(3S)
an existing one. creat:	create a new file or rewrite . . . . .	creat(2)
fork:	create a new process. . . . .	fork(2)
tmpfile:	create a temporary file. . . . .	tmpfile(3S)
communication. socket:	create an endpoint for . . . . .	socket(2N)
by massaging C source. mkstr:	create an error message file . . . . .	mkstr(1)
channel. pipe:	create an interprocess . . . . .	pipe(2)
files. admin:	create and administer SCCS . . . . .	admin(1)
umask: set and get file	creation mask. . . . .	umask(2)
cribbage: the card game	cribbage. . . . .	cribbage(6)
cribbage.	cribbage: the card game . . . . .	cribbage(6)
	cron: clock daemon. . . . .	cron(1M)
crontab: user	crontab file. . . . .	crontab(1)
	crontab: user crontab file. . . . .	crontab(1)
cxref: generate C program	cross-reference. . . . .	cxref(1)
optimization package. curses:	CRT screen handling and . . . . .	curses(3X)
more: file perusal filter for	crt viewing. . . . .	more(1)
	crypt: encode/decode. . . . .	crypt(1)
generate DES encryption.	crypt, setkey, encrypt: . . . . .	crypt(3C)
interpreter) with C-like/	csk: a shell (command . . . . .	csk(1)
function. sin, dsin,	csin: Fortran sine intrinsic . . . . .	sin(3F)
	csplit: context split. . . . .	csplit(1)
intrinsic/ sqrt, dsqrt,	csqrt: Fortran square root . . . . .	sqrt(3F)
terminal.	ct: spawn getty to a remote . . . . .	ct(1C)
for a C program.	ctags: maintain a tags file . . . . .	ctags(1)
terminal.	ctermid: generate filename for . . . . .	ctermid(3S)
asctime, tzset: convert date/	ctime, localtime, gmtime, . . . . .	ctime(3C)
	ctrace: C program debugger. . . . .	ctrace(1)
	cu: call another UNIX system. . . . .	cu(1C)
ttt,	cubic: tic-tac-toe. . . . .	ttt(6)
get/set unique identifier of	current host. /sethostid: . . . . .	gethostid(2N)
sethostname: get/set name of	current host. gethostname, . . . . .	gethostname(2N)
set or print identifier of	current host system. hostid: . . . . .	hostid(1N)
hostname: set or print name of	current host system. . . . .	hostname(1N)
activity. sact: print	current SCCS file editing . . . . .	sact(1)
chgnod: change	current UNIX system nodename. . . . .	chgnod(1M)
uname: print name of	current UNIX system. . . . .	uname(1)
uname: get name of	current UNIX system. . . . .	uname(2)
whoami: print effective	current user id. . . . .	whoami(1)
slot in the utmp file of the	current user. /find the . . . . .	ttyslot(3C)
getcwd: get pathname of	current working directory. . . . .	getcwd(3C)
and optimization package.	curses: CRT screen handling . . . . .	curses(3X)
spline: interpolate smooth	curve. . . . .	spline(1G)
name of the user.	userid: get character login . . . . .	userid(3S)
of each line of a file.	cut: cut out selected fields . . . . .	cut(1)
each line of a file. cut:	cut out selected fields of . . . . .	cut(1)
constant-width text for/	cw, checkcw: prepare . . . . .	cw(1)

cross-reference.	cxref: generate C program . . . . .	cxref(1)
absolute value. abs, labs,	dabs, cabs, zabs: Fortran . . . . .	abs(3F)
intrinsic function. acos,	dacos: Fortran arccosine . . . . .	acos(3F)
cron: clock	daemon. . . . .	cron(1M)
errdemon: error-logging	daemon. . . . .	errdemon(1M)
terminate the error-logging	daemon. errstop: . . . . .	errstop(1M)
routed: network routing	daemon. . . . .	routed(8N)
runacct: run	daily accounting. . . . .	runacct(1M)
backup. filesave, tapesave:	daily/weekly UNIX file system	filesave(1M)
Protocol server. ftpd:	DARPA Internet File Transfer . . . . .	ftpd(8N)
telnetd:	DARPA TELNET protocol server. . . . .	telnetd(8N)
Protocol server. tftpd:	DARPA Trivial File Transfer . . . . .	tftpd(8N)
/handle special functions of	DASI 300 and 300s terminals. . . . .	300(1)
special functions of the	DASI 450 terminal. /handle . . . . .	450(1)
intrinsic function. asin,	dasin: Fortran arcsine . . . . .	asin(3F)
/time a command; report process	data and system activity. . . . .	timex(1)
hosts: host name	data base. . . . .	hosts(4N)
networks: network name	data base. . . . .	networks(4N)
pört. ttytype:	data base of terminal types by . . . . .	ttytype(4)
protocols: protocol name	data base. . . . .	protocols(4N)
services: service name	data base. . . . .	services(4N)
termcap: terminal capability	data base. . . . .	termcap(5)
terminfo: terminal capability	data base. . . . .	terminfo(4)
blt, blt512: block transfer	data. . . . .	blt(3C)
generate disk accounting	data by user ID. diskusg: . . . . .	diskusg(1M)
/sgetl: access long integer	data in a machine independent/	sputl(3X)
plock: lock process, text, or	data in memory. . . . .	plock(2)
prof: display profile	data. . . . .	prof(1)
call. stat:	data returned by stat system . . . . .	stat(5)
brk, sbrk: change	data segment space allocation. . . . .	brk(2)
types: primitive system	data types. . . . .	types(5)
join: relational	database operator. . . . .	join(1)
tput: query terminfo	database. . . . .	tput(1)
udp: Internet User	Datagram Protocol. . . . .	udp(5P)
intrinsic function. atan,	datan: Fortran arctangent . . . . .	atan(3F)
intrinsic function. atan2,	datan2: Fortran arctangent . . . . .	atan2(3F)
/asctime, tzset: convert	date and time to string. . . . .	ctime(3C)
date: print and set the	date. . . . .	date(1)
/idint, real, float, sngl,	date: print and set the date. . . . .	date(1)
/float, sngl, dble, cmplx,	dble, cmplx, dcmplx, ichar,/	fype(3F)
conjugate intrinsic/ conjg,	dc: desk calculator. . . . .	dc(1)
optimal access time.	dcmplx, ichar, char: explicit/	fype(3F)
intrinsic function. cos,	dconjg: Fortran complex . . . . .	conjg(3F)
cosine intrinsic/ cosh,	dcopy: copy file systems for . . . . .	dcopy(1M)
	dcos, ccos: Fortran cosine . . . . .	cos(3F)
	dcosh: Fortran hyperbolic . . . . .	cosh(3F)
	dd: convert and copy a file. . . . .	dd(1)
difference intrinsic/ dim,	ddim, idim: positive . . . . .	dim(3F)
ctrace: C program	debugger. . . . .	ctrace(1)
fsdb: file system	debugger. . . . .	fsdb(1M)
sdb: symbolic	debugger. . . . .	sdb(1)
sysdef: system	definition. . . . .	sysdef(1M)
eqnchar: special character	definitions for eqn and neqn. . . . .	eqnchar(5)
people. delivermail:	deliver mail to arbitrary . . . . .	delivermail(8N)
netmailer:	deliver mail to B-NET. . . . .	netmailer(8N)
names. basename, dirname:	deliver portions of path . . . . .	basename(1)
file. tail:	deliver the last part of a . . . . .	tail(1)
aliases: aliases file for	delivermail. . . . .	aliases(7N)
arbitrary people.	delivermail: deliver mail to . . . . .	delivermail(8N)
delta commentary of an SCCS	delta. cdc: change the . . . . .	cdc(1)
file. delta: make a	delta (change) to an SCCS . . . . .	delta(1)
delta. cdc: change the	delta commentary of an SCCS . . . . .	cdc(1)
rmddl: remove a	delta from an SCCS file. . . . .	rmddl(1)
to an SCCS file.	delta: make a delta (change) . . . . .	delta(1)

comb: combine SCCS	deltas.	comb(1)
msg: permit or	deny messages.	msg(1)
tbl, and eqn constructs.	deroff: remove nroff/troff,	deroff(1)
setkey, encrypt: generate	DES encryption. crypt,	crypt(3C)
close: close a file	descriptor.	close(2)
dup2: duplicate a	descriptor.	dup2(3N)
dup: duplicate a	descriptor.	dup(3)
getdtablesize: get	descriptor table size.	getdtablesize(3N)
dc:	desk calculator.	dc(1)
file. access:	determine accessibility of a	access(2)
file:	determine file type.	file(1)
errors in the specified	device. /on/off the extended	exterr(1)
master: master	device information table.	master(4)
ioctl: control	device.	ioctl(2)
devnm:	device name.	devnm(1M)
devnm: device name.	devnm: device name.	devnm(1M)
exponential intrinsic/ exp,	dexp, cexp: Fortran	exp(3F)
blocks.	df: report number of free disk	df(1M)
check and interactive/ fsck,	dfck: file system consistency	fsck(1M)
terminal line connection.	dial: establish an out-going	dial(3C)
ratfor: rational Fortran	dialect.	ratfor(1)
bdiff: big	diff.	bdiff(1)
comparator.	diff: differential file	diff(1)
diffdir:	diff directories.	diffdir(1)
comparison.	diff3: 3-way differential file	diff3(1)
diffdir: diff directories.	diffdir: diff directories.	diffdir(1)
dim, ddim, idim: positive	difference intrinsic/	dim(3F)
sdiff: side-by-side	difference program.	sdiff(1)
diffmk: mark	differences between files.	diffmk(1)
diff:	differential file comparator.	diff(1)
diff3: 3-way	differential file comparison.	diff3(1)
between files.	diffmk: mark differences	diffmk(1)
difference intrinsic/	dim, ddim, idim: positive	dim(3F)
of complex argument. aimag,	dimag: Fortran imaginary part	aimag(3F)
intrinsic function. aint,	dint: Fortran integer part	aint(3F)
dir: format of directories.	dir: format of directories.	dir(4)
diremp: directory comparison.	diremp: directory comparison.	diremp(1)
directories. cpset:	directories. cpset:	cpset(1M)
directories.	directories.	diffdir(1)
directories.	directories.	dir(4)
directories.	directories.	rm(1)
directories. /count characters	directories. /count characters	sumdir(1)
directory.	directory.	cd(1)
directory.	directory.	chdir(2)
directory.	directory.	chroot(2)
directory clean-up.	directory clean-up.	uuclean(1M)
directory comparison.	directory comparison.	diremp(1)
directory entry.	directory entry.	unlink(2)
directory for a command.	directory for a command.	chroot(1M)
directory for fsck.	directory for fsck.	mklost+found(1M)
directory. getcwd: get	directory. getcwd: get	getcwd(3C)
directory.	directory.	ls(1)
directory.	directory.	mkdir(1)
directory.	directory.	mkdir(1M)
directory name.	directory name.	pwd(1)
directory operations.	directory operations.	directory(3X)
directory, or a special or	directory, or a special or	mkknod(2)
dirname: deliver portions of	dirname: deliver portions of	basename(1)
dis: disassembler.	dis: disassembler.	dis(1)
disable: enable/disable LP	disable: enable/disable LP	enable(1)
disable process accounting.	disable process accounting.	acct(2)
disassembler.	disassembler.	dis(1)
discipline. /set terminal	discipline. /set terminal	getty(1M)
disk accounting data by user	disk accounting data by user	diskusg(1M)

df: report number of free disk blocks. . . . . df(1M)  
 diskformat: format a disk. . . . . diskformat(1M)  
 disktime: tune floppy disk settling time parameters. . . . . disktime(1M)  
 du: summarize disk usage. . . . . du(1)  
 diskformat: format a disk. . . . . diskformat(1M)  
 disktime: tune floppy disk . . . . . disktime(1M)  
 diskusg: generate disk usage data by user ID. . . . . diskusg(1M)  
 dismount, umount: mount and dismount file system. . . . . mount(1M)  
 vi: screen-oriented (visual) display editor based on ex. . . . . vi(1)  
 prof: display profile data. . . . . prof(1)  
 rain: animated raindrops. . . . . rain(6)  
 worms: animate worms on a display terminal. . . . . worms(6)  
 hypot: Euclidean distance function. . . . . hypot(3M)  
 /icong48: generate uniformly distributed pseudo-random/ dlog, clog: Fortran natural . . . . . drand48(3C)  
 logarithm/ log, alog, dlog10: Fortran common . . . . . log(3F)  
 logarithm/ log10, alog10, dmax1: Fortran maximum-value/ . . . . . log10(3F)  
 max, max0, amax0, max1, amax1, dmin1: Fortran minimum-value/ . . . . . max(3F)  
 min, min0, amin0, min1, amin1, dmod: Fortran remaindering . . . . . min(3F)  
 intrinsic/ mod, amod, dnint, nint, idnint: Fortran . . . . . mod(3F)  
 nearest integer/ anint, documents formatted with the/ . . . . . round(3F)  
 mm, osdd, checkmm: print/check macro package for formatting documents. mm: the MM . . . . . mm(1)  
 macro package for formatting slides. mmt, mvt: typeset . . . . . mm(5)  
 nulladm,/ chargefee, ckpaoct, documents. /the OSDD adapter . . . . . mosd(5)  
 whoho: who is mmt(1)  
 intrinsic function. dprod: acctsh(1M)  
 /atof: convert string to whodo(1M)  
 /Motorola S-records from dprod(3F)  
 product intrinsic function. strtod(3C)  
 nrand48, mrand48, jrand48,/ rcvhex(1)  
 graph. dprod(3F)  
 arithmetic: provide drand48(3C)  
 pt: pseudo terminal graph(1G)  
 sxt: pseudo-device arithmetic(6)  
 transfer-of-sign/ sign, isign, pt(5)  
 intrinsic function. sin, sign(3F)  
 intrinsic function. sinh, sin(3F)  
 root intrinsic/ sqrt, sinh(3F)  
 intrinsic function. tan, sqrt(3F)  
 tangent intrinsic/ tanh, tan(3F)  
 tanh(3F)  
 du(1)  
 an object file. dump: dump selected parts of . . . . . dump(1)  
 extract error records from dump. errdead: . . . . . errdead(1M)  
 od: octal dump. . . . . od(1)  
 object file. dump: dump selected parts of an . . . . . dump(1)  
 dup: duplicate a descriptor. . . . . dup(3)  
 dup2: duplicate a descriptor. . . . . dup2(3N)  
 duplicate a descriptor. . . . . dup2(3N)  
 dup: duplicate a descriptor. . . . . dup(3)  
 The alien invaders attack the earth. aliens: . . . . . aliens(6)  
 echo: echo arguments. . . . . echo(1)  
 echo arguments. . . . . echo(1)  
 floating-point number to/ ecvt, fcvt, gcvt: convert . . . . . ecvt(3C)  
 program. end, etext, ed, red: text editor. . . . . ed(1)  
 ex, edit: text editor. . . . . end(3C)  
 edit: text editor. . . . . ex(1)  
 editing activity. . . . . sact(1)  
 editor based on ex. . . . . vi(1)  
 editor. . . . . ed(1)  
 editor. . . . . ex(1)  
 editor for common object files. ld: link . . . . . ld(1)  
 ld5.0: link . . . . . ld5.0(1)  
 common assembler and link editor output. a.out: . . . . . a.out(4)



a.out5.0: assembler and link sed: stream whoami: print setregid: set real and /user, real group, and setreuid: set real and and/ /getegid: get real user. Language. fsplit: split f77, ratfor, or for a pattern. grep, insque, remque: insert/remove enable/disable LP printers. accounting. acct: enable, disable: crypt: encryption. crypt, setkey, setkey, encrypt: generate DES makekey: generate locations in program. /getgrgid, getgrnam, setgrent, /gethostbyname, sethostent, /getnetbyname, setnetent, socket: create an /getprotobyname, setprotoent, /getpwuid, getpwnam, setpwent, /getservbyname, setservent, utmp/ /pututline, setutent, convert Arabic numerals to nlist: get file. linenum: line number man: print man: macros for formatting file/ /manipulate line number /ldnseek: seek to line number /kdnrseek: seek to relocation utmp, wtmp: utmp and wtmp /fgetgrnt: obtain group file endhostent: get network host endnetent: get network endprotoent: get protocol fgetpwent: get password file endservent: get service utmpname: access utmp file /the index of a symbol table /read an indexed symbol table putpwent: write password file unlink: remove directory command execution.  profile: setting up an environ: user execution. env: set getenv: return value for printenv: print out the putenv: change or add value to getenv: return Fortran character definitions for remove nroff/troff, tbl, and mathematical text for nroff/ definitions for eqn and neqn. mrand48, jrand48, / drand48, complementary error function. complementary error/ erf, from dump.	editor output. editor. effective current user id. effective group ID. effective group IDs. effective user ID's. effective user, real group, efl: Extended Fortran efl files. egrep, fgrep: search a file element from a queue. enable, disable: enable or disable process enable/disable LP printers. encode/decode. encrypt: generate DES encryption. crypt, encryption key. end, etext, edata: last endgrent, fgetgrent: obtain/ endhostent: get network host/ endnetent: get network entry. endpoint for communication. endprotoent: get protocol/ endpwent, fgetpwent: get/ endservent: get service entry. endutent, utmpname: access English. number: entries from name list. entries in a common object entries in this manual. entries in this manual. entries of a common object entries of a section of a/ entries of a section of a/ entry formats. entry from a group file. entry. /sethostent, entry. /setnetent, entry. /setprotoent, entry. /setpwent, endpwent, entry. /setservent, entry. /setutent, endutent, entry of a common object file. entry of a common object file. entry. entry. env: set environment for environ: user environment. environment at login time. environment. environment for command environment name. environment. environment. environment variable. eqn and neqn. /special eqn constructs. deroff: eqn, neqn, checkeq: format eqnchar: special character erand48, lrand48, nrand48, erf, erfc: error function and erfc: error function and errdead: extract error records	a.out5.0(4) sed(1) whoami(1) setregid(2) getuid(2) setreuid(2) getuid(2) efl(1) fsplit(1) grep(1) insque(3N) enable(1) acct(2) enable(1) crypt(1) crypt(3C) crypt(3C) makekey(1) end(3C) getgrent(3C) gethostent(3N) getnetent(3N) socket(2N) getprotoent(3N) getpwent(3C) getservent(3N) getut(3C) number(6) nlist(3C) linenum(4) man(1) man(5) ldread(3X) ldseek(3X) ldrseek(3X) utmp(4) getgrent(3C) gethostent(3N) getnetent(3N) getprotoent(3N) getpwent(3C) getservent(3N) getut(3C) ldtindex(3X) ldtread(3X) putpwent(3C) unlink(2) env(1) environ(5) profile(4) environ(5) env(1) getenv(3C) printenv(1) putenv(3C) getenv(3F) eqnchar(5) deroff(1) eqn(1) eqnchar(5) drand48(3C) erf(3M) erf(3M) errdead(1M)
--	---	--

daemon. errdemon: error-logging . . . . . errdemon(1M)  
 format. errfile: error-log file . . . . . errfile(4)  
 system error/ perror, error(7)  
 interface. error: error-logging . . . . . error(7)  
 complementary/ erf, erfc: error function and . . . . . erf(3M)  
 function and complementary erf(3M)  
 messaging C/ mkstr: create an error message file by . . . . . mkstr(1)  
 sys\_errlist, sys\_nerr: system error messages. /errno, . . . . . perror(3C)  
 to system calls and error numbers. /introduction . . . . . intro(2)  
 errdead: extract error records from dump. . . . . errdead(1M)  
 matherr: error-handling function. . . . . matherr(3M)  
 errfile: error-log file format. . . . . errfile(4)  
 errdemon: error-logging daemon. . . . . errdemon(1M)  
 errstop: terminate the error-logging daemon. . . . . errstop(1M)  
 error: error-logging interface. . . . . error(7)  
 process a report of logged errors. errpt: . . . . . errpt(1M)  
 /turn on/off the extended errors in the specified/ . . . . . exterr(1)  
 hashcheck: find spelling errors. /hashmake, spellin, . . . . . spell(1)  
 logged errors. errpt: process a report of . . . . . errpt(1M)  
 error-logging daemon. errstop: terminate the . . . . . errstop(1M)  
 robots. autorobots: Escape from the automatic . . . . . autorobots(6)  
 robots: Escape from the robots. . . . . robots(6)  
 chase: Try to escape the killer robots. . . . . chase(6)  
 terminal line/ dial: establish an out-going . . . . . dial(3C)  
 setmnt: establish mount table. . . . . setmnt(1M)  
 in program. end, edata: last locations . . . . . end(3C)  
 hypot: Euclidean distance function. . . . . hypot(3M)  
 expression. expr: evaluate arguments as an . . . . . expr(1)  
 test: condition evaluation command. . . . . test(1)  
 ex, edit: text editor. . . . . ex(1)  
 display editor based on ex. /screen-oriented (visual) . . . . . vi(1)  
 reading or/ locking: provide exclusive file regions for . . . . . locking(2)  
 execlp, execvp: execute a/ execl, execv, execl, execl, . . . . . exec(2)  
 execvp: execute/ execl, execv, execl, execl, . . . . . exec(2)  
 execl, execv, execl, execl, . . . . . exec(2)  
 execl, execv, execl, execl, . . . . . exec(2)  
 regcmp, regex: compile and execute a regular expression. . . . . regcmp(3X)  
 construct argument list(s) and execute command. xargs: . . . . . xargs(1)  
 time. at, batch: execute commands at a later . . . . . at(1)  
 set environment for command execution. env: . . . . . env(1)  
 sleep: suspend execution for an interval. . . . . sleep(1)  
 sleep: suspend execution for interval. . . . . sleep(3C)  
 monitor: prepare execution profile. . . . . monitor(3C)  
 rexecd: remote execution server. . . . . rexecd(8N)  
 profil: execution time profile. . . . . profil(2)  
 UNIX-to-UNIX system command execution. uux: . . . . . uux(1C)  
 execvp: execute a/ execl, execv, execl, execl, . . . . . exec(2)  
 execl, execl, execv, execl, . . . . . exec(2)  
 /execv, execl, execl, . . . . . exec(2)  
 system calls. link, unlink: exercise link and unlink . . . . . link(1M)  
 a new file or rewrite an existing one. creat: create . . . . . creat(2)  
 process. exit, \_exit: terminate . . . . . exit(2)  
 exit, \_exit: terminate process. . . . . exit(2)  
 exponential intrinsic/ exp, dexp, cexp: Fortran . . . . . exp(3F)  
 exponential, logarithm,/ exp, log, log10, pow, sqrt: . . . . . exp(3M)  
 pcat, unpack: compress and expand files. pack, . . . . . pack(1)  
 cmplx, dcmplx, ichar, char: explicit Fortran type/ /dble, . . . . . ftype(3F)  
 adventure: an exploration game. . . . . adventure(6)  
 exp, dexp, cexp: Fortran exponential intrinsic/ . . . . . exp(3F)  
 exp, log, log10, pow, sqrt: exponential, logarithm, power,/ . . . . . exp(3M)  
 expression. expr: evaluate arguments as an . . . . . expr(1)  
 routines. regex: regular expression compile and match . . . . . regex(5)  
 regcmp: regular expression compile. . . . . regcmp(1)  
 expr: evaluate arguments as an expression. . . . . expr(1)

compile and execute a regular	expression, regcomp, regex: . . . . .	regcomp(3X)
exterr: turn on/off the	extended errors in the/ . . . . .	exterr(1)
efl:	Extended Fortran Language. . . . .	efl(1)
greek: graphics for the	extended TTY-37 type-box. . . . .	greek(5)
extended errors in the/	exterr: turn on/off the . . . . .	exterr(1)
dump. errdead:	extract error records from . . . . .	errdead(1M)
programs to implement/ xstr:	extract strings from C . . . . .	xstr(1)
	f77: Fortran 77 compiler. . . . .	f77(1)
	f77, ratfor, or efl files. . . . .	fsplit(1)
fsplit: split	fabs: floor, ceiling, . . . . .	floor(3M)
remainder,/ floor, ceil, fmod,	factor a number. . . . .	factor(1)
factor:	factor: factor a number. . . . .	factor(1)
true,	false: provide truth values. . . . .	true(1)
data in a machine independent	fashion. /access long integer . . . . .	spuil(3X)
finc:	fast incremental backup. . . . .	finc(1M)
/calloc, malloc, mallinfo:	fast main memory allocator. . . . .	malloc(3X)
procedure. checkall:	faster file system checking . . . . .	checkall(1M)
abort: generate an IOT	fault. . . . .	abort(3C)
a stream.	fclose, fflush: close or flush . . . . .	fclose(3S)
	fcntl: file control. . . . .	fcntl(2)
	fcntl: file control options. . . . .	fcntl(5)
floating-point number/ cvt,	fcvt, gcvt: convert . . . . .	cvcl(3C)
fopen, freopen,	fdopen: open a stream. . . . .	fopen(3S)
status inquiries. ferror,	feof, clearerr, fileno: stream . . . . .	ferror(3S)
fileno: stream status/	ferror, feof, clearerr, . . . . .	ferror(3S)
statistics for a file system.	ff: list file names and . . . . .	ff(1M)
stream. fclose,	flush: close or flush a . . . . .	fclose(3S)
bcopy, bcmp, bzero,	ffs: bit and byte string/ . . . . .	bstring(3N)
word from a/getc, getchar,	fgetc, getw: get character or . . . . .	getc(3S)
/getgrnam, setgrent, endgrent,	fgetgrent: obtain group file/ . . . . .	getgrent(3C)
/getpwnam, setpwent, endpwent,	fgetpwent: get password file/ . . . . .	getpwent(3C)
stream. gets,	fgets: get a string from a . . . . .	gets(3S)
pattern. grep, egrep,	fgrep: search a file for a . . . . .	grep(1)
times. utime: set	file access and modification . . . . .	utime(2)
lfcn: common object	file access routines. . . . .	ldfcn(4)
determine accessibility of a	file. access: . . . . .	access(2)
tar: tape	file archiver. . . . .	tar(1)
cpio: copy	file archives in and out. . . . .	cpio(1)
mkstr: create an error message	file by massaging C source. . . . .	mkstr(1)
pwck, grpck: password/group	file checkers. . . . .	pwck(1M)
chmod: change mode of	file. . . . .	chmod(2)
change owner and group of a	file. chown: . . . . .	chown(2)
diff: differential	file comparator. . . . .	diff(1)
diff3: 3-way differential	file comparison. . . . .	diff3(1)
fcntl:	file control. . . . .	fcntl(2)
fcntl:	file control options. . . . .	fcntl(5)
conv: object	file converter. . . . .	conv(1)
rcp: remote	file copy. . . . .	rcp(1N)
public UNIX-to-UNIX system	file copy. uuto, uupick: . . . . .	uuto(1C)
core: format of core image	file. . . . .	core(4)
umask: set and get	file creation mask. . . . .	umask(2)
crontab: user crontab	file. . . . .	crontab(1)
fields of each line of a	file. cut: cut out selected . . . . .	cut(1)
dd: convert and copy a	file. . . . .	dd(1)
a delta (change) to an SCCS	file. delta: make . . . . .	delta(1)
close: close a	file descriptor. . . . .	close(2)
	file: determine file type. . . . .	file(1)
selected parts of an object	file. dump: dump . . . . .	dump(1)
sact: print current SCCS	file editing activity. . . . .	sact(1)
fgetgrent: obtain group	file entry from a group file. . . . .	getgrent(3C)
fgetpwent: get password	file entry. /endpwent, . . . . .	getpwent(3C)
utmpname: access utmp	file entry. /endutent, . . . . .	getut(3C)
putpwent: write password	file entry. . . . .	putpwent(3C)
execvp, execvp: execute a	file. /execv, execl, execl, . . . . .	exec(2)

ctags: maintain a tags	file for a C program. . . . .	ctags(1)
grep, egrep, fgrep: search a	file for a pattern. . . . .	grep(1)
aliases: aliases	file for delivermail. . . . .	aliases(7N)
ldaoopen: open a common object	file for reading. ldopen. . . . .	ldopen(3X)
acct: per-process accounting	file format. . . . .	acct(4)
ar: common archive	file format. . . . .	ar(4)
ar5.0: archive (library)	file format. . . . .	ar5.0(4)
errfile: error-log	file format. . . . .	errfile(4)
pnch:	file format for card images. . . . .	pnch(4)
intro: introduction to	file formats. . . . .	intro(4)
on character frequencies in a	file. freq: report . . . . .	freq(1)
take: takes a	file from a remote machine. . . . .	take(1C)
entries of a common object	file function. /line number . . . . .	ldread(3X)
get: get a version of an SCCS	file. . . . .	get(1)
group file entry from a group	file. /fgetgrent: obtain . . . . .	getgrent(3C)
group: group	file. . . . .	group(4)
files. filehdr:	file header for common object . . . . .	filehdr(4)
file. ldhread: read the	file header of a common object . . . . .	ldhread(3X)
ldohseek: seek to the optional	file header of a common object/ . . . . .	ldohseek(3X)
split: split a	file into pieces. . . . .	split(1)
issue: issue identification	file. . . . .	issue(4)
of a member of an archive	file. /read the archive header . . . . .	ldahread(3X)
close a common object	file. ldclose, ldclose: . . . . .	ldclose(3X)
file header of a common object	file. ldhread: read the . . . . .	ldhread(3X)
symbol name for object	file. ldgetname: retrieve . . . . .	ldgetname(3X)
a section of a common object	file. /line number entries of . . . . .	ldlseek(3X)
file header of a common object	file. /seek to the optional . . . . .	ldohseek(3X)
a section of a common object	file. /relocation entries of . . . . .	ldrseek(3X)
header of a common object	file. /indexed/named section . . . . .	ldhseek(3X)
section of a common object	file. /to an indexed/named . . . . .	ldsseek(3X)
table entry of a common object	file. /the index of a symbol . . . . .	ldtbindex(3X)
table entry of a common object	file. /read an indexed symbol . . . . .	ldtbread(3X)
table of a common object	file. /seek to the symbol . . . . .	ldtseek(3X)
entries in a common object	file. linenum: line number . . . . .	linenum(4)
link: link to a	file. . . . .	link(2)
mknod: build special	file. . . . .	mknod(1M)
or a special or ordinary	file. /make a directory. . . . .	mknod(2)
a file system. ff: list	file names and statistics for . . . . .	ff(1M)
change the format of a text	file. newform: . . . . .	newform(1)
name list of common object	file. nm: print . . . . .	nm(1)
null: the null	file. . . . .	null(7)
/find the slot in the utmp	file of the current user. . . . .	ttyslot(3C)
put: puts a	file onto a remote machine. . . . .	put(1C)
/identify processes using a	file or file structure. . . . .	fuser(1M)
one. creat: create a new	file or rewrite an existing . . . . .	creat(2)
passwd: password	file. . . . .	passwd(4)
or subsequent lines of one	file. /lines of several files . . . . .	paste(1)
viewing. more:	file perusal filter for crt . . . . .	more(1)
soft-copy terminals. pg:	file perusal filter for . . . . .	pg(1)
/rewind, ftall: reposition a	file pointer in a stream. . . . .	fseek(3S)
lseek: move read/write	file pointer. . . . .	lseek(2)
prs: print an SCCS	file. . . . .	prs(1)
from downloading into a	file. /Motorola S-records . . . . .	rcvhex(1)
read: read from	file. . . . .	read(2)
ready: read from	file. . . . .	readv(3N)
locking: provide exclusive	file regions for reading or/ . . . . .	locking(2)
for a common object	file. /relocation information . . . . .	reloc(4)
remove a delta from an SCCS	file. rmdel: . . . . .	rmdel(1)
bfs: big	file scanner. . . . .	bfs(1)
two versions of an SCCS	file. sccsdiff: compare . . . . .	sccsdiff(1)
sccsfile: format of SCCS	file. . . . .	sccsfile(4)
header for a common object	file. scnhdr: section . . . . .	scnhdr(4)
size5.0: size of an object	file. . . . .	size5.0(1)
stat, fstat: get	file status. . . . .	stat(2)

in an object, or other binary information from an object	file. /the printable strings	strings(1)
processes using a file or checksum and block count of a sum and count blocks in a syms: common object	file. /symbol and line number	strip(1)
tapesave: daily/weekly UNIX procedure. checkall: faster and interactive/ fsck, dfck: fsdb: names and statistics for a volume.	file structure. /identify	fuser(1M)
mkfslb: construct a mkfs: construct a umount: mount and dismount mount: mount a ustat: get mnttab: mounted umount: unmount a access time. dcopy: copy fsck. checklist: list of volcopy, labelit: copy deliver the last part of a term: format of compiled term tmpfile: create a temporary create a name for a temporary and modification times of a ftp: ftpd: DARPA Internet tfpd: DARPA Trivial ftw: walk a file: determine undo a previous get of an SCCS report repeated lines in a val: validate SCCS write: write on a writev: write on a umask: set common object files. ctermid: generate mktemp: make a unique ferror, feof, clearerr, and print process accounting merge or add total accounting create and administer SCCS a.out header for common object VAX-11/780/ fscv: convert updater: update updater: update cat: concatenate and print cmp: compare two lines common to two sorted cp, ln, mv: copy, link or move mark differences between file header for common object find: find frec: recover format specification in text split f77, raufor, or elf hex: translates object cpset: install object and count characters in the intro: introduction to special link editor for common object	file. sum: print	sum(1)
	file. sum7:	sum7(1)
	file symbol table format.	syms(4)
	file system backup. filesave,	filesave(1M)
	file system checking	checkall(1M)
	file system consistency check	fsck(1M)
	file system debugger.	fsdb(1M)
	file system. ff: list file	ff(1M)
	file system: format of system	fs(4)
	file system.	mkfslb(1M)
	file system.	mkfs(1M)
	file system. mount,	mount(1M)
	file system.	mount(2)
	file system statistics.	ustat(2)
	file system table.	mnttab(4)
	file system.	umount(2)
	file systems for optimal	dcopy(1M)
	file systems processed by	checklist(4)
	file systems with label/	volcopy(1M)
	file. tail:	tail(1)
	file..	term(4)
	file.	tmpfile(3S)
	file. tmpnam, tempnam:	tmpnam(3S)
	file. touch: update access	touch(1)
	file transfer program.	ftp(1N)
	File Transfer Protocol server.	ftpd(8N)
	File Transfer Protocol server.	tfpd(8N)
	file tree.	ftw(3C)
	file type.	file(1)
	file. unget:	unget(1)
	file. uniq:	uniq(1)
	file.	val(1)
	file.	write(3)
	file.	writev(3N)
	file-creation mode mask.	umask(1)
	filehdr: file header for	filehdr(4)
	filename for terminal.	ctermid(3S)
	filename.	mktemp(3C)
	fileno: stream status/	ferror(3S)
	file(s). acctcorn: search	acctcom(1)
	files. acctmerg:	acctmerg(1M)
	files. admin:	admin(1)
	files. aouthdr.h	aouthdr(4)
	files between M68000 and	fscv(1M)
	files between two machines.	updater(1)
	files between two machines.	updater(1M)
	files.	cat(1)
	files.	cmp(1)
	files. comm: select or reject	comm(1)
	files.	cp(1)
	files. diffmk:	diffmk(1)
	files. filehdr:	filehdr(4)
	files.	find(1)
	files from a backup tape.	frec(1M)
	files. fspec:	fspec(4)
	files. fsplit:	fsplit(1)
	files.	hex(1)
	files in binary directories.	cpset(1M)
	files in the given/ /sum	sumdir(1)
	files.	intro(7)
	files. ld:	ld(1)

lockf: record locking on	files.	lockf(3C)
rm, rmdir: remove	files or directories.	rm(1)
/merge same lines of several	files or subsequent lines of/	paste(1)
unpack: compress and expand	files. pack, pcat,	pack(1)
pr: print	files.	pr(1)
section sizes of common object	files. size: print	size(1)
sort: sort and/or merge	files.	sort(1)
reports version number of	files. version:	version(1)
what: identify SCCS	files.	what(1)
daily/weekly UNIX file system/	filesave, tapesave:	filesave(1M)
more: file perusal	filter for crt viewing.	more(1)
terminals. pg: file perusal	filter for soft-copy	pg(1)
greek: select terminal	filter.	greek(1)
nl: line numbering	filter.	nl(1)
col:	filter reverse line-feeds.	col(1)
tplot: graphics	filters.	tplot(1G)
	finc: fast incremental backup.	finc(1M)
find:	find files.	find(1)
	find: find files.	find(1)
hyphen:	find hyphenated words.	hyphen(1)
ttyname, isatty:	find name of a terminal.	ttyname(3C)
object library. lorder:	find ordering relation for an	lorder(1)
object library. lorder5.0:	find ordering relation for an	lorder5.0(1)
hashmake, spellin, hashcheck:	find spelling errors. spell.	spell(1)
an object, or other/ strings:	find the printable strings in	strings(1)
of the current user. ttyslot:	find the slot in the utmp file	ttyslot(3C)
fish: play "Go	Fish".	fish(6)
tee: pipe	fish: play "Go Fish".	fish(6)
/seekdir, rewinddir, closedir:	fitting.	tee(1)
int, ifix, idint, real,	flexible length directory/	directory(3X)
atof: convert ASCII string to	float, snl, dbl, cmplx./	ftype(3F)
ecvt, fcvt, gcvt: convert	floating-point number.	atof(3C)
/modf: manipulate parts of	floating-point number to/	ecvt(3C)
floor, ceiling, remainder./	floating-point numbers.	frexp(3C)
floor, ceil, fmod, fabs:	floor, ceil, fmod, fabs:	floor(3M)
parameters. disktime: tune	floor, ceiling, remainder./	floor(3M)
cflow: generate C	floppy disk settling time	disktime(1M)
fclose, fflush: close or	flowgraph.	cflow(1)
remainder./ floor, ceil,	flush a stream.	fclose(3S)
stream.	fmod, fabs: floor, ceiling,	floor(3M)
	fopen, freopen, fdopen: open a	fopen(3S)
diskformat:	fork: create a new process.	fork(2)
per-process accounting file	format a disk.	diskformat(1M)
ar: common archive file	format. acct:	acct(4)
ar5.0: archive (library) file	format.	ar(4)
errfile: error-log file	format.	ar5.0(4)
pch: file	format.	errfile(4)
nroff or/ eqn, neqn, checkeq:	format for card images.	pch(4)
newform: change the	format mathematical text for	eqn(1)
inode:	format of a text file.	newform(1)
term:	format of an inode.	inode(4)
core:	format of compiled term file..	term(4)
cpio:	format of core image file.	core(4)
dir:	format of cpio archive.	cpio(4)
sccsfile:	format of directories.	dir(4)
file system:	format of SCCS file.	sccsfile(4)
files. fspec:	format of system volume.	fs(4)
object file symbol table	format specification in text	fspec(4)
troff. tbl:	format. syms: common	syms(4)
nroff:	format tables for nroff or	tbl(1)
intro: introduction to file	format text.	nroff(1)
wtmp: utmp and wtmp entry	formats.	intro(4)
scanf, fscanf, sscanf: convert	formats. utmp,	utmp(4)
	formatted input.	scanf(3S)

/vfprintf, vsprintf: print	formatted output of a varargs/	vfprintf(3S)
/vfprintf, vsprintf: print	formatted output of a varargs/	vfprintf(3X)
fprintf, sprintf: print	formatted output. printf,	printf(3S)
/checkmm: print/check documents	formatted with the MM macros.	mm(1)
mpix: the macro package for	formatting a permuted index.	mpix(5)
mm: the MM macro package for	formatting documents.	mm(5)
OSDD adapter macro package for	formatting documents. /the	mosd(5)
manual. man: macros for	formatting entries in this	man(5)
f77:	Fortran 77 compiler.	f77(1)
abs, iabs, dabs, cabs, zabs:	Fortran absolute value.	abs(3F)
system/ signal: specify	Fortran action on receipt of a	signal(3F)
function. acos, dacos:	Fortran arccosine intrinsic	acos(3F)
function. asin, dasin:	Fortran arcsine intrinsic	asin(3F)
function. atan2, datan2:	Fortran arctangent intrinsic	atan2(3F)
function. atan, datan:	Fortran arctangent intrinsic	atan(3F)
or, xor, not, lshift, rshift:	Fortran bitwise boolean/ and,	bool(3F)
getarg: return	Fortran command-line argument.	getarg(3F)
log10, alog10, dlog10:	Fortran common logarithm/	log10(3F)
intrinsic/ conjg, dconjg:	Fortran complex conjugate	conjg(3F)
function. cos, dcos, ccos:	Fortran cosine intrinsic	cos(3F)
function. ratfor: rational	Fortran dialect.	ratfor(1)
function. getenv: return	Fortran environment variable.	getenv(3F)
function. exp, dexp, cexp:	Fortran exponential intrinsic	exp(3F)
intrinsic/ cosh, dcosh:	Fortran hyperbolic cosine	cosh(3F)
intrinsic/ sinh, dsinh:	Fortran hyperbolic sine	sinh(3F)
intrinsic/ tanh, dtanh:	Fortran hyperbolic tangent	tanh(3F)
complex/ aimag, dimag:	Fortran imaginary part of	aimag(3F)
function. aint, dint:	Fortran integer part intrinsic	aint(3F)
efl: Extended	Fortran Language.	efl(1)
amax0, max1, amax1, dmax1:	Fortran maximum-value/ /max0,	max(3F)
amin0, min1, amin1, dmin1:	Fortran minimum-value/ /min0,	min(3F)
log, alog, dlog, clog:	Fortran natural logarithm/	log(3F)
anint, dnint, nint, idnint:	Fortran nearest integer/	round(3F)
abort: terminate	Fortran program.	abort(3F)
functions. mod, amod, dmod:	Fortran remaindering intrinsic	mod(3F)
function. sin, dsin, csin:	Fortran sine intrinsic	sin(3F)
function. sqrt, dsqrt, csqrt:	Fortran square root intrinsic	sqrt(3F)
len: return length of	Fortran string.	len(3F)
index: return location of	Fortran substring.	index(3F)
issue a shell command from	Fortran. system:	system(3F)
function. tan, dtan:	Fortran tangent intrinsic	tan(3F)
function. mclock: return	Fortran time accounting.	mclock(3F)
intrinsic/ sign, isign, dsign:	Fortran transfer-of-sign	sign(3F)
/dcmplx, ichar, char: explicit	Fortran type conversion.	ftype(3F)
irand, srand, rand:	Fortran uniform random-number/	rand(3F)
hopefully interesting, adage:	fortune: print a random,	fortune(6)
formatted output. printf,	fprintf, sprintf: print	printf(3S)
word on a/ putc, putchar,	fputc, putw: put character or	putc(3S)
stream. puts,	fputs: put a string on a	puts(3S)
input/output.	fread, fwrite: binary	fread(3S)
backup tape.	frec: recover files from a	frec(1M)
df: report number of	free disk blocks.	df(1M)
memory allocator. malloc,	free, realloc, calloc: main	malloc(3C)
malloc, malinfo:/ malloc,	free, realloc, calloc,	malloc(3X)
stream. fopen,	freopen, fdopen: open a	fopen(3S)
frequencies in a file.	freq: report on character	freq(1)
freq: report on character	frequencies in a file.	freq(1)
parts of floating-point/	frexp, ldexp, modf: manipulate	frexp(3C)
frec: recover files	from a backup tape.	frec(1M)
obtain group file entry	from a group file. /fgetgrent:	getgrent(3C)
remque: insert/remove element	from a queue. insque,	insque(3N)
take: takes a file	from a remote machine.	take(1C)
recvmsg: receive a message	from a socket. /recvfrom,	recv(2N)
sendmsg: send a message	from a socket. send, sendto,	send(2N)

getw: get character or word	from a stream. /fgetc, . . . . .	getc(3S)
gets, fgets: get a string	from a stream. . . . .	gets(3S)
and line number information	from an object file. /symbol . . . . .	strip(1)
rmdbl: remove a delta	from an SCCS file. . . . .	rmdbl(1)
getopt: get option letter	from argument vector. . . . .	getopt(3C)
shared/ xstr: extract strings	from C programs to implement . . . . .	xstr(1)
/translates Motorola S-records	from downloading into a file. . . . .	rcvhex(1)
errdead: extract error records	from dump. . . . .	errdead(1M)
read: read	from file. . . . .	read(2)
readv: read	from file. . . . .	readv(3N)
system: issue a shell command	from Fortran. . . . .	system(3F)
ncheck: generate names	from i-numbers. . . . .	ncheck(1M)
nlist: get entries	from name list. . . . .	nlist(3C)
acctcms: command summary	from per-process accounting/ . . . . .	acctcms(1M)
autorobots: Escape	from the automatic robots. . . . .	autorobots(6)
robots: Escape	from the robots. . . . .	robots(6)
getpw: get name	from UID. . . . .	getpw(3C)
formatted input. scanf,	fscanf, sscanf: convert . . . . .	scanf(3S)
of file systems processed by	fsock. checklist: list . . . . .	checklist(4)
consistency check and/	fsock, dfsock: file system . . . . .	fsock(1M)
a lost+found directory for	fsock. mklost+found: make . . . . .	mklost+found(1M)
M68000 and VAX-11/780/	fscv: convert files between . . . . .	fscv(1M)
reposition a file pointer in/	fsdb: file system debugger. . . . .	fsdb(1M)
text files.	fseek, rewind, ftell: . . . . .	fseek(3S)
efl files.	fspec: format specification in . . . . .	fspec(4)
stat.	fsplit: split f77, ratfor, or . . . . .	fsplit(1)
pointer in a/ fseek, rewind,	fstat: get file status. . . . .	stat(2)
communication package.	ftell: reposition a file . . . . .	fseek(3S)
Transfer Protocol server.	ftok: standard interprocess . . . . .	stidpc(3C)
shutdown: shut down part of a	ftp: file transfer program. . . . .	ftp(1N)
Fortran arccosine intrinsic	ftpd: DARPA Internet File . . . . .	ftpd(8N)
Fortran integer part intrinsic	ftw: walk a file tree. . . . .	ftw(3C)
error/ erf, erfc: error	full-duplex connection. . . . .	shutdown(2N)
Fortran arcsine intrinsic	function. acos, dacos: . . . . .	acos(3F)
Fortran arctangent intrinsic	function. aint, dint: . . . . .	aint(3F)
complex conjugate intrinsic	function and complementary . . . . .	erf(3M)
ccos: Fortran cosine intrinsic	function. asin, dasin: . . . . .	asin(3F)
hyperbolic cosine intrinsic	function. atan2, datan2: . . . . .	atan2(3F)
precision product intrinsic	function. atan, datan: . . . . .	atan(3F)
and complementary error	function. /dconj: Fortran . . . . .	conjg(3F)
Fortran exponential intrinsic	function. cos, dcos, . . . . .	cos(3F)
gamma: log gamma	function. /dcosh: Fortran . . . . .	cosh(3F)
hypot: Euclidean distance	function. dprod: double . . . . .	dprod(3F)
of a common object file	function. /error function . . . . .	erf(3M)
common logarithm intrinsic	function. exp, dexp, cexp: . . . . .	exp(3F)
natural logarithm intrinsic	function. . . . .	gamma(3M)
matherr: error-handling	function. /line number entries . . . . .	hypot(3M)
prof: profile within a	function. /dlog10: Fortran . . . . .	ldread(3X)
transfer-of-sign intrinsic	function. /dlog, clog: Fortran . . . . .	log10(3F)
csin: Fortran sine intrinsic	function. . . . .	log(3F)
hyperbolic sine intrinsic	function. /dsign: Fortran . . . . .	matherr(3M)
Fortran square root intrinsic	function. sin, dsin, . . . . .	prof(5)
Fortran tangent intrinsic	function. /dsinh: Fortran . . . . .	sign(3F)
hyperbolic tangent intrinsic	function. sqrt, dsqrt, csqrt: . . . . .	sin(3F)
math: math	function. tan, dtan: . . . . .	sinh(3F)
j0, j1, jn, y0, y1, yn: Bessel	function. /dtanh: Fortran . . . . .	sqrt(3F)
Fortran bitwise boolean	functions and constants. . . . .	tan(3F)
positive difference intrinsic	functions. /lshift, rshift: . . . . .	tanh(3F)
logarithm, power, square root	functions. dim, ddim, idim: . . . . .	math(5)
remainder, absolute value	functions. /sqrt: exponential, . . . . .	bessel(3M)
	functions. /floor, ceiling, . . . . .	bool(3F)
		exp(3M)
		floor(3M)



dmax1: Fortran maximum-value	functions. /max1, amax1, . . . . .	max(3F)
dmin1: Fortran minimum-value	functions. /min1, amin1, . . . . .	min(3F)
Fortran remaindering intrinsic	functions. mod, amod, dmod: . . . . .	mod(3F)
300, 300s: handle special	functions of DASI 300 and 300s/ . . . . .	300(1)
terminal. 450: handle special	functions of the DASI 450 . . . . .	450(1)
Fortran nearest integer	functions. /nint, idnint: . . . . .	round(3F)
sinh, cosh, tanh: hyperbolic	functions. . . . .	sinh(3M)
string comparison intrinsic	functions. /lgt, lle, llt: . . . . .	strempr(3F)
atan, atan2: trigonometric	functions. /tan, asin, acos, . . . . .	trig(3M)
using a file or file/	fuser: identify processes . . . . .	fuser(1M)
read,	fwrite: binary input/output. . . . .	fread(3S)
connect accounting records.	fwtmp, wtmpfix: manipulate . . . . .	fwtmp(1M)
adventure: an exploration	game. . . . .	adventure(6)
cribbage: the card	game cribbage. . . . .	cribbage(6)
moo: guessing	game. . . . .	moo(6)
back: the	game of backgammon. . . . .	back(6)
bj: the	game of black jack. . . . .	bj(6)
craps: the	game of craps. . . . .	craps(6)
wump: the	game of hunt-the-wumpus. . . . .	wump(6)
life: play the	game of life. . . . .	life(6)
trek: trekkie	game. . . . .	trek(6)
worm: Play the growing worm	game. . . . .	worm(6)
intro: introduction to	games. . . . .	intro(6)
gamma: log	gamma function. . . . .	gamma(3M)
	gamma: log gamma function. . . . .	gamma(3M)
number to string. ecvt, fcvt,	gcvt: convert floating-point . . . . .	ecvt(3C)
maze:	generate a maze. . . . .	maze(6)
abort:	generate an IOT fault. . . . .	abort(3C)
cflow:	generate C flowgraph. . . . .	cflow(1)
cross-reference. cxref:	generate C program . . . . .	cxref(1)
crypt, setkey, encrypt:	generate DES encryption. . . . .	crypt(3C)
by user ID. diskusg:	generate disk accounting data . . . . .	diskusg(1M)
makekey:	generate encryption key. . . . .	makekey(1)
terminal. ctermid:	generate filename for . . . . .	ctermid(3S)
ncheck:	generate names from i-numbers. . . . .	ncheck(1M)
lexical tasks. lex:	generate programs for simple . . . . .	lex(1)
/srand48, seed48, lcong48:	generate uniformly distributed/ . . . . .	drand48(3C)
srand: simple random-number	generator. rand, . . . . .	rand(3C)
Fortran uniform random-number	generator. /srand, rand: . . . . .	rand(3F)
gets, fgets:	get a string from a stream. . . . .	gets(3S)
get:	get a version of an SCCS file. . . . .	get(1)
getsockopt, setsockopt:	get and set options on/ . . . . .	getsockopt(2N)
ulimit:	get and set user limits. . . . .	ulimit(2)
the user. cuserid:	get character login name of . . . . .	cuserid(3S)
getc, getchar, fgetc, getw:	get character or word from a/ . . . . .	getc(3S)
getdtablesize:	get descriptor table size. . . . .	getdtablesize(3N)
nlist:	get entries from name list. . . . .	nlist(3C)
umask: set and	get file creation mask. . . . .	umask(2)
stat, fstat:	get file status. . . . .	stat(2)
ustat:	get file system statistics. . . . .	ustat(2)
file.	get: get a version of an SCCS . . . . .	get(1)
getlogin:	get login name. . . . .	getlogin(3C)
logname:	get login name. . . . .	logname(1)
msgget:	get message queue. . . . .	msgget(2)
getpw:	get name from UID. . . . .	getpw(3C)
getpeername:	get name of connected peer. . . . .	getpeername(2N)
system. uname:	get name of current UNIX . . . . .	uname(2)
/setnetent, endnetent:	get network entry. . . . .	getnetent(3N)
/sethostent, endhostent:	get network host entry. . . . .	gethostent(3N)
unget: undo a previous	get of an SCCS file. . . . .	unget(1)
argument vector. getopt:	get option letter from . . . . .	getopt(3C)
/setpwent, endpwent, fgetpwent:	get password file entry. . . . .	getpwent(3C)
working directory. getcwd:	get pathname of current . . . . .	getcwd(3C)
times. times:	get process and child process . . . . .	times(2)

and/ getpid, getpgrp, getppid: get process, process group, . . . . . getpid(2)  
 /setprotoent, endprotoent: get protocol entry. . . . . getprotoent(3N)  
 /geteuid, getgid, getegid: get real user, effective user./ . . . . . getuid(2)  
 /setservent, endservent: get service entry. . . . . setservent(3N)  
 semget: get set of semaphores. . . . . semget(2)  
 shmget: get shared memory segment. . . . . shmget(2)  
 getsockname: get socket name. . . . . getsockname(2N)  
 tty: get the terminal's name. . . . . tty(1)  
 time: get time. . . . . time(2)  
 command-line argument. getarg: return Fortran . . . . . getarg(3F)  
 get character or word from a/ getc, getch, fgetc, getw: . . . . . getc(3S)  
 character or word from/ getc, getch, fgetc, getw: get . . . . . getc(3S)  
 current working directory. getcwd: get pathname of . . . . . getcwd(3C)  
 table size. getdtablesize: get descriptor . . . . . getdtablesize(3N)  
 getuid, geteuid, getgid, getegid: get real user./ . . . . . getuid(2)  
 environment variable. getenv: return Fortran . . . . . getenv(3F)  
 environment name. getenv: return value for . . . . . getenv(3C)  
 real user, effective/ getuid, geteuid, getgid, getegid: get . . . . . getuid(2)  
 user,/ getuid, geteuid, getgid, getegid: get real . . . . . getuid(2)  
 setgrent, endgrent,/ getgrent, getgrgid, getgrnam, setgrent, . . . . . getgrent(3C)  
 endgrent,/ getgrent, getgrgid, getgrnam, setgrent, . . . . . getgrent(3C)  
 getgrent, getgrgid, getgrent, endgrent,/ getgrent(3C)  
 sethostent,/ gethostent, gethostbyaddr, gethostbyname, sethostent,/ . . . . . gethostent(3N)  
 gethostent, gethostbyaddr, gethostbyname, sethostent,/ . . . . . gethostent(3N)  
 gethostbyname, sethostent,/ gethostent, gethostbyaddr, . . . . . gethostent(3N)  
 unique identifier of current/ gethostid, sethostid: get/set . . . . . gethostid(2N)  
 get/set name of current host. gethostname, sethostname: . . . . . gethostname(2N)  
 getlogin: get login name. . . . . getlogin(3C)  
 setnetent,/ getnetent, getnetbyaddr, getnetbyname, setnetent,/ . . . . . getnetent(3N)  
 getnetbyname, setnetent,/ getnetent, getnetbyaddr, . . . . . getnetent(3N)  
 argument vector. getopt: get option letter from . . . . . getopt(3C)  
 getopt: parse command options. . . . . getopt(1)  
 getpass: read a password. . . . . getpass(3C)  
 connected peer. getpeername: get name of . . . . . getpeername(2N)  
 process group, and/ getpid, process group, and/ getpgrp, getppid: get process, . . . . . getpid(2)  
 group, and/ getpid, getpgrp, process group, and/ getpgrp, . . . . . getpid(2)  
 getprotoent, getprotobyname, getprotobyname,/ setprotoent, . . . . . getprotoent(3N)  
 getprotobyname,/ setprotoent, getprotobyname, . . . . . getprotoent(3N)  
 getprotoent, getprotobyname, setprotoent,/ getprotoent(3N)  
 getpw: get name from UID. . . . . getpw(3C)  
 setpwent, endpwent,/ getpwent, getpwuid, getpwnam, setpwent, endpwent,/ . . . . . getpwent(3C)  
 endpwent,/ getpwent, getpwuid, getpwnam, setpwent, . . . . . getpwent(3C)  
 a stream. gets, fgets: get a string from . . . . . gets(3S)  
 getservent, getservbyport, getservbyname, setservent,/ . . . . . getservent(3N)  
 setservent,/ getservent, getservbyport, getservbyname, . . . . . getservent(3N)  
 getservbyname, setservent,/ getservent, getservbyport, . . . . . getservent(3N)  
 gethostname, sethostname: get/set name of current host. . . . . gethostname(2N)  
 current/ gethostid, sethostid: get/set unique identifier of . . . . . gethostid(2N)  
 getsockname: get socket name. . . . . getsockname(2N)  
 and set options on sockets. getsockopt, setsockopt: get . . . . . getsockopt(2N)  
 and terminal settings used by getty. getty. gettydefs: speed . . . . . gettydefs(4)  
 modes, speed, and line/ getty: set terminal type, . . . . . getty(1M)  
 ct: spawn getty to a remote terminal. . . . . ct(1C)  
 settings used by getty. gettydefs: speed and terminal . . . . . gettydefs(4)  
 getuid, geteuid, getgid, putline, setutent,/ . . . . . getuid(2)  
 putline, setutent,/ getutent, getutid, getutline, . . . . . getut(3C)  
 setutent, endutent,/ getutent, getutid, getutline, putline, . . . . . getut(3C)  
 setutent,/ getutent, getutid, getutline, putline, . . . . . getut(3C)  
 from a/ getc, getch, fgetc, getw: get character or word . . . . . getc(3S)  
 convert/ ctime, localtime, gmtime, asctime, tzset: . . . . . ctime(3C)  
 fish: play "Go Fish". . . . . fish(6)

setjmp, longjmp: non-local	goto	setjmp(3C)
graph: draw a	graph: draw a graph	graph(1G)
sag: system activity	graph	graph(1G)
tplot:	graphics filters	sag(1G)
TTY-37 type-box. greek:	graphics for the extended	tplot(1G)
plot:	graphics interface	greek(5)
subroutines. plot:	graphics interface	plot(4)
mvt: typeset documents, view	graphs, and slides. mmt,	plot(3X)
package for typesetting view	graphs and slides. /macro	mmt(1)
extended TTY-37 type-box.	greek: graphics for the	mv(5)
file for a pattern.	greek: select terminal filter.	greek(5)
/user, effective user, real	grep, egrep, fgrep: search a	greek(1)
/getppid: get process, process	group, and effective group/	grep(1)
chown, chgrp: change owner or	group, and parent process IDs.	getuid(2)
/ndgrent, fgetgrent: obtain	group.	getpid(2)
obtain group file entry from	group file entry from a group/	chown(1)
group:	group file. /fgetgrent:	getgrent(3C)
setpgrp: set process	group file.	getgrent(3C)
set real and effective	group: group file	group(4)
id: print user and	group ID.	group(4)
real group, and effective	group ID. setregid:	setpgrp(2)
setuid, setgid: set user and	group IDs and names.	setregid(2)
send signal to a process	group IDs. /effective user,	id(1)
newgrp: log in to a new	group IDs.	getuid(2)
chown: change owner and	group. killpg:	setuid(2)
a signal to a process or a	group.	killpg(3N)
update, and regenerate	group of a file.	newgrp(1)
worm: Play the	group of processes. /send	chown(2)
checkers. pwck,	groups of programs. /maintain,	kill(2)
signal,	growing worm game.	make(1)
hangman:	grpck: password/group file	worm(6)
moo:	gsignal: software signals.	pwck(1M)
DASI 300 and 300s/ 300, 300s:	guess the word.	ssignal(3C)
the DASI 450 terminal. 450:	guessing game.	hangman(6)
varargs:	handle special functions of	moo(6)
information for bad block	handle special functions of	300(1)
package. curses: CRT screen	handle variable argument list.	450(1)
nohup: run a command immune to	handling. /alternate block	varargs(5)
hcreate, hdestroy: manage	handling and optimization	altblk(4)
spell, hashmake, spellin,	hangman: guess the word.	curses(3X)
find spelling errors. spell,	hangups (sh only).	hangman(6)
search tables. hsearch,	hash search tables. hsearch,	nohup(1)
tables. hsearch, hcreate,	hashcheck: find spelling/	hsearch(3C)
file. scnhdr: section	hashmake, spellin, hashcheck:	spell(1)
files. aouthdr.h - a.out	hcreate, hdestroy: manage hash	hsearch(3C)
files. filehdr: file	hdestroy: manage hash search	hsearch(3C)
file. ldhread: read the file	header for a common object	scnhdr(4)
/seek to the optional file	header for common object	aouthdr(4)
/read an indexed/named section	header for common object	filehdr(4)
ldahread: read the archive	header of a common object	ldhread(3X)
SCCS.	header of a common object/	ldohseek(3X)
help: ask for	header of a common object/	ldhread(3X)
fortune: print a random,	header of a member of an/	ldahread(3X)
/ntohs: convert values between	help: ask for help in using	help(1)
endhostent: get network	help in using SCCS.	help(1)
unique identifier of current	hex: translates object files.	hex(1)
get/set name of current	hopefully interesting. adage.	fortune(6)
hosts:	host and network byte order.	byteorder(3N)
ruptime: show	host entry. /sethostent,	gethostent(3N)
	host. /sethostid: get/set	gethostid(2N)
	host. /sethostname:	gethostname(2N)
	host name data base.	hosts(4N)
	host status of local machines.	ruptime(1N)

*Permuted Index*

or print identifier of current set or print name of current identifier of current host/ current host system.	host system, hostid: set . . . . .	hostid(1N)
	host system, hostname: . . . . .	hostname(1N)
	hostid: set or print . . . . .	hostid(1N)
	hostname: set or print name of . . . . .	hostname(1N)
	hosts: host name data base. . . . .	hosts(4N)
manage hash search tables.	hsearch, hcreate, hdestroy: . . . . .	hsearch(3C)
convert values between host/ values between host/ htonl, htols, ntohl, ntohs: wump: the game of	htonl, htols, ntohl, ntohs: . . . . .	byteorder(3N)
cosh, dcosh: Fortran	hunt-the-wumpus. . . . .	wump(6)
sinh, cosh, tanh: sinh, dsinh: Fortran	hyperbolic cosine intrinsic/ . . . . .	cosh(3F)
tanh, dtanh: Fortran	hyperbolic functions. . . . .	sinh(3M)
	hyperbolic sine intrinsic/ . . . . .	sinh(3F)
	hyperbolic tangent intrinsic/ . . . . .	tanh(3F)
	hyphen: find hyphenated words. . . . .	hyphen(1)
hyphen: find function.	hyphenated words. . . . .	hyphen(1)
Fortran absolute value. abs,	hypot: Euclidean distance . . . . .	hypot(3M)
	iabs, dabs, cabs, zabs: . . . . .	abs(3F)
	iargc: . . . . .	iargc(3F)
/sngl, dble, cmplx, dcmplx, disk accounting data by user	ichar, char: explicit Fortran/ . . . . .	ftype(3F)
semaphore set or shared memory and names.	ID. diskusg: generate . . . . .	diskusg(1M)
setpgrp: set process group set real and effective group print effective current user issue: issue	id. /remove a message queue. . . . .	ipcrm(1)
/sethostid: get/set unique system. hostid: set or print file or file/ fuser: what: intrinsic/ dim, ddim, dble, cmplx,/ int, fix, integer/ anint, dnint, nint, id: print user and group group, and parent process group, and effective group set real and effective user segid: set user and group interface parameters. sngl, dble, cmplx,/ int, core: format of core pncb: file format for card aimag, dimag: Fortran nohup: run a command /strings from C programs to finc: fast	id: print user and group IDs . . . . .	id(1)
long integer data in a machine /goto, tputs: terminal for formatting a permuted of a/ ldtbindex: compute the ptx: permuted Fortran substring. a common/ ldtbread: read an ldsbread, ldsnbread: read an ldsseek, ldsseek: seek to an and teletypes. last: family.	ID. . . . .	setpgrp(2)
	ID. setregid: . . . . .	setregid(2)
	id. whoami: . . . . .	whoami(1)
	identification file. . . . .	issue(4)
	identifier of current host. . . . .	gethostid(2N)
	identifier of current host . . . . .	hostid(1N)
	identify processes using a identify SCCS files. . . . .	fuser(1M)
	idim: positive difference . . . . .	what(1)
	idint, real, float, sngl. . . . .	dim(3F)
	idnint: Fortran nearest IDs and names. . . . .	ftype(3F)
	IDs. /get process, process . . . . .	round(3F)
	IDs. /effective user, real . . . . .	id(1)
	ID's. setreuid: . . . . .	getpid(2)
	IDs. setuid, . . . . .	getuid(2)
	ifconfg: configure network . . . . .	setreuid(2)
	ifix, idint, real, float, . . . . .	setuid(2)
	image file. . . . .	ifconfg(8N)
	images. . . . .	ftype(3F)
	imaginary part of complex/ . . . . .	core(4)
	immune to hangups (sh only). . . . .	core(4)
	implement shared strings. . . . .	pncb(4)
	incremental backup. . . . .	aimag(3F)
	independent fashion. /access . . . . .	nohup(1)
	independent operation/ . . . . .	xstr(1)
	index. /the macro package . . . . .	finc(1M)
	index of a symbol table entry . . . . .	sputt(3X)
	index. . . . .	termcap(3X)
	index: return location of . . . . .	mptx(5)
	indexed symbol table entry of . . . . .	ldtbindex(3X)
	indexed/named section header/ . . . . .	ptx(1)
	indexed/named section of a/ . . . . .	index(3F)
	indicate last logins of users . . . . .	ldtbread(3X)
	inet: Internet protocol . . . . .	ldshread(3X)
	inet_addr, inet_network, . . . . .	ldsseek(3X)
	inet_lnaof, inet_netof:/ . . . . .	last(1)
	inet_makeaddr, inet_lnaof,/ . . . . .	inet(5F)
	inet_netof: Internet address/ . . . . .	inet(3N)
	inet_network, inet_ntoa, . . . . .	inet(3N)
	inet_ntoa, inet_makeaddr,/ . . . . .	inet(3N)
	init process. . . . .	inet(3N)
		inittab(4)

initialization. . . . . init(1M)  
 init, telinit: process control . . . . . init(1M)  
 /rc, powerfail: system . . . . . brc(1M)  
   socket. connect: . . . . . connect(2N)  
 process. popen, pclose: . . . . . popen(3S)  
   process. . . . . initab(4)  
   cli: clear . . . . . cli(1M)  
   inode: format of an . . . . . inode(4)  
 sscanf: convert formatted . . . . . inode(4)  
 push character back into . . . . . scanf(3S)  
   fread, fwrite: binary . . . . . ungetc(3S)  
 stdio: standard buffered . . . . . fread(3S)  
   fileno: stream status . . . . . stdio(3S)  
   ustat: uucp status . . . . . ferror(3S)  
 queue. insque, remque: . . . . . uustat(1C)  
   element from a queue. . . . . insque(3N)  
   install: . . . . . insque(3N)  
   install commands. . . . . install(1M)  
   install: install commands. . . . . install(1M)  
   install object files in binary . . . . . cpset(1M)  
 directories. cpset: . . . . . ftype(3F)  
 singl, dble, cmplx, dcmplx, / . . . . . abs(3C)  
   abs: return . . . . . a64i(3C)  
 /l64a: convert between long . . . . . sputl(3X)  
   sputl, agel: access long . . . . . round(3F)  
 nint, idnint: Fortran nearest . . . . . aint(3F)  
 function. aint, dint: Fortran . . . . . strtol(3C)  
 atol, atoi: convert string to . . . . . l3tol(3C)  
 /lto13: convert between 3-byte . . . . . l3tol(3C)  
   3-byte integers and long . . . . . bcopy(1M)  
   bcopy: . . . . . mailx(1)  
   system. mailx: . . . . . fsck(1M)  
 system consistency check and . . . . . fortune(6)  
   print a random, hopefully . . . . . error(7)  
   error: error-logging . . . . . lo(5)  
 lo: software loopback network . . . . . ifconfig(8N)  
   ifconfig: configure network . . . . . plot(4)  
   plot: graphics . . . . . plot(3X)  
   plot: graphics . . . . . termio(7)  
 termio: general terminal . . . . . telnet(1N)  
   protocol. telnet: user . . . . . tty(7)  
   tty: controlling terminal . . . . . inet(3N)  
   /inet\_inaof, inet\_netof: . . . . . ftpd(8N)  
 Protocol server. ftpd: DARPA . . . . . inet(5F)  
   inet: . . . . . ip(5P)  
   ip: . . . . . tcp(5P)  
   Protocol. tcp: . . . . . udp(5P)  
   Protocol. udp: . . . . . spline(1G)  
   spline: . . . . . asa(1)  
   characters. asa: . . . . . sno(1)  
   sno: SNOBOL . . . . . csh(1)  
 syntax. csh: a shell (command . . . . . pipe(2)  
   pipe: create an . . . . . ipcs(1)  
   facilities/ ipcs: report . . . . . stdipc(3C)  
   package. ftok: standard . . . . . sleep(1)  
   suspend execution for an . . . . . sleep(3C)  
   sleep: suspend execution for . . . . . acos(3F)  
 acos, dacos: Fortran arccosine . . . . . aint(3F)  
 dint: Fortran integer part . . . . . asin(3F)  
 asin, dasin: Fortran arcsine . . . . . atan2(3F)  
 datan2: Fortran arctangent . . . . . atan(3F)  
 datan: Fortran arctangent . . . . . conjg(3F)  
 Fortran complex conjugate . . . . . cos(3F)  
 dcos, ccos: Fortran cosine . . . . . cosh(3F)  
 Fortran hyperbolic cosine . . . . .

double precision product	intrinsic function. dprod: . . . . .	dprod(3F)
cexp: Fortran exponential	intrinsic function. /dexp, . . . . .	exp(3F)
Fortran common logarithm	intrinsic function. /dlog10: . . . . .	log10(3F)
Fortran natural logarithm	intrinsic function. /dlog: . . . . .	log(3F)
Fortran transfer-of-sign	intrinsic function. /dsign: . . . . .	sign(3F)
sin, dsin, csin: Fortran sine	intrinsic function. . . . .	sin(3F)
dsinh: Fortran hyperbolic sine	intrinsic function. sinh, . . . . .	sinh(3F)
csqrt: Fortran square root	intrinsic function. /dsqrt, . . . . .	sqrt(3F)
tan, dtan: Fortran tangent	intrinsic function. . . . .	tan(3F)
Fortran hyperbolic tangent	intrinsic function. /dtanh: . . . . .	tanh(3F)
idim: positive difference	intrinsic functions. /ddim, . . . . .	dim(3F)
dmod: Fortran remaindering	intrinsic functions. /amod, . . . . .	mod(3F)
lle, llt: string comparison	intrinsic functions. /lgt, . . . . .	strcmp(3F)
commands and application/ formats.	intro: introduction to . . . . .	intro(1)
	intro: introduction to file . . . . .	intro(4)
	intro: introduction to games. . . . .	intro(6)
	intro: introduction to . . . . .	intro(5)
miscellany.	intro: introduction to special . . . . .	intro(7)
files.	intro: introduction to . . . . .	intro(3)
subroutines and libraries.	intro: introduction to system . . . . .	intro(2)
calls and error numbers.	intro: introduction to system . . . . .	intro(1M)
maintenance commands and/ maintenance procedures.	intro: introduction to system . . . . .	intro(8)
application programs. intro:	introduction to commands and . . . . .	intro(1)
	introduction to file formats. . . . .	intro(4)
	intro: introduction to games. . . . .	intro(6)
	intro: introduction to miscellany. . . . .	intro(5)
facilities. networking:	introduction to networking . . . . .	intro(5N)
	intro: introduction to special files. . . . .	intro(7)
and libraries. intro:	introduction to subroutines . . . . .	intro(3)
and error numbers. intro:	introduction to system calls . . . . .	intro(2)
maintenance commands/ intro:	introduction to system . . . . .	intro(1M)
maintenance/ intro:	introduction to system . . . . .	intro(8)
ncheck: generate names from	i-numbers. . . . .	ncheck(1M)
aliens: The alien	invaders attack the earth. . . . .	aliens(6)
select: synchronous	i/o multiplexing. . . . .	select(2N)
	ioctl: control device. . . . .	ioctl(2)
abort: generate an	IOT fault. . . . .	abort(3C)
	ip: Internet Protocol. . . . .	ip(5P)
semaphore set or shared/ communication facilities/ uniform random-number/	ipcrm: remove a message queue, . . . . .	ipcrm(1)
/islower, isdigit, isxdigit,	ipcs: report inter-process . . . . .	ipcs(1)
isdigit, isxdigit, isalnum,/	irand, srand, rand: Fortran . . . . .	rand(3F)
/isprint, isgraph, isctrl,	isalnum, ispace, ispunct,/	ctype(3C)
terminal. ttyname,	isalpha, isupper, islower, . . . . .	ctype(3C)
/ispunct, isprint, isgraph,	isascii: classify characters. . . . .	ctype(3C)
isalpha, isupper, islower,	isatty: find name of a . . . . .	ttyname(3C)
/isspace, ispunct, isprint,	isctrl, isascii: classify/ . . . . .	ctype(3C)
transfer-of-sign/ sign,	isdigit, isxdigit, isalnum,/	ctype(3C)
isalnum,/ isalpha, isupper,	isgraph, isctrl, isascii/ . . . . .	ctype(3C)
/isalnum, ispace, ispunct,	isign, dsign: Fortran . . . . .	sign(3F)
/isxdigit, isalnum, ispace,	islower, isdigit, isxdigit, . . . . .	ctype(3C)
/isdigit, isxdigit, isalnum,	isprint, isgraph, isctrl,/	ctype(3C)
Fortran. system:	ispunct, isprint, isgraph,/	ctype(3C)
system:	isspace, ispunct, isprint,/	ctype(3C)
issue:	issue a shell command from . . . . .	system(3F)
file.	issue a shell command. . . . .	system(3S)
isxdigit, isalnum,/ isalpha,	issue identification file. . . . .	issue(4)
/isupper, islower, isdigit,	issue: issue identification . . . . .	issue(4)
news: print news	isupper, islower, isdigit, . . . . .	ctype(3C)
functions.	isxdigit, isalnum, ispace,/	ctype(3C)
functions. j0,	items. . . . .	news(1)
bj: the game of black	j0, j1, jn, y0, y1, yn: Bessel . . . . .	bessel(3M)
functions. j0, j1,	j1, jn, y0, y1, yn: Bessel . . . . .	bessel(3M)
	jack. . . . .	bj(6)
	jn, y0, y1, yn: Bessel . . . . .	bessel(3M)

operator.	join: relational database . . . . .	join(1)
/rand48, nrand48, mrand48,	jrnd48, srnd48, seed48./	drand48(3C)
makekey: generate encryption	key. . . . .	makekey(1)
killall:	kill all active processes. . . . .	killall(1M)
process or a group of/	kill: send a signal to a . . . . .	kill(2)
	kill: terminate a process. . . . .	kill(1)
processes.	killall: kill all active . . . . .	killall(1M)
chase: Try to escape the	killer robots. . . . .	chase(6)
process group.	killpg: send signal to a . . . . .	killpg(3N)
mem,	kmem: core memory. . . . .	mem(7)
quiz: test your	knowledge. . . . .	quiz(6)
3-byte integers and long/	l3tol, l0l3: convert between . . . . .	l3tol(3C)
integer and base-64/ a64l,	l64a: convert between long . . . . .	a64l(3C)
copy file systems with	label checking. /labelit:	volcopy(1M)
label checking. volcopy:	labelit: copy file systems . . . . .	volcopy(1M)
scanning and processing	language. awk: pattern . . . . .	awk(1)
arbitrary-precision arithmetic	language. bc: . . . . .	bc(1)
efl: Extended Fortran	Language. . . . .	efl(1)
cpp: the C	language preprocessor. . . . .	cpp(1)
cpp: the C	language preprocessor. . . . .	cpp5.0(1)
command programming	language. /standard/restricted . . . . .	sh(1)
chargefee, ckpacct, dodisk,	lastlogin, monacct, nulladm./	acctsh(1M)
statistics.	lav: print load average . . . . .	lav(1)
shl: shell	layer manager. . . . .	shl(1)
/jrnd48, srnd48, seed48,	lcong48: generate uniformly/	drand48(3C)
object files.	ld: link editor for common . . . . .	ld(1)
	ld5.0: link editor. . . . .	ld5.0(1)
object file. ldclose:	ldaclose: close a common . . . . .	ldclose(3X)
header of a member of an/	ldahread: read the archive . . . . .	ldahread(3X)
file for reading. ldopen:	ldaopen: open a common object . . . . .	ldopen(3X)
common object file.	ldclose, ldaclose: close a . . . . .	ldclose(3X)
of floating-point/ frexp,	ldexp, modf: manipulate parts . . . . .	frexp(3C)
access routines.	ldfcn: common object file . . . . .	ldfcn(4)
of a common object file.	ldfhread: read the file header . . . . .	ldfhread(3X)
name for object file.	ldgetname: retrieve symbol . . . . .	ldgetname(3X)
line number entries/ ldread,	ldlinit, ldliitem: manipulate . . . . .	ldlread(3X)
number/ ldread, ldliitem,	ldliitem: manipulate line . . . . .	ldlread(3X)
manipulate line number/	ldlread, ldliinit, ldliitem: . . . . .	ldlread(3X)
line number entries of a/	ldlseek, kdnlseek: seek to . . . . .	ldlseek(3X)
entries of a section/ ldseek,	ldnlseek: seek to line number . . . . .	ldseek(3X)
entries of a section/ ldseek,	ldnrseek: seek to relocation . . . . .	ldrseek(3X)
indexed/named/ ldshread,	ldnshread: read an . . . . .	ldshread(3X)
indexed/named/ ldsseek,	ldnsseek: seek to an . . . . .	ldsseek(3X)
file header of a common/	ldohseek: seek to the optional . . . . .	ldohseek(3X)
object file for reading.	ldopen, ldaopen: open a common . . . . .	ldopen(3X)
relocation entries of a/	ldrseek, ldnrseek: seek to . . . . .	ldrseek(3X)
indexed/named section header/	ldshread, ldnsbread: read an . . . . .	ldshread(3X)
indexed/named section of a/	ldsseek, ldnsseek: seek to an . . . . .	ldsseek(3X)
of a symbol table entry of a/	ldtbindx: compute the index . . . . .	ldtbindx(3X)
symbol table entry of a/	ldtbread: read an indexed . . . . .	ldtbread(3X)
table of a common object/	ldtbseek: seek to the symbol . . . . .	ldtbseek(3X)
string.	len: return length of Fortran . . . . .	len(3F)
/rewinddir, closedir: flexible	length directory operations. . . . .	directory(3X)
len: return	length of Fortran string. . . . .	len(3F)
getopt: get option	letter from argument vector. . . . .	getopt(3C)
simple lexical tasks.	lex: generate programs for . . . . .	lex(1)
generate programs for simple	lexical tasks. lex: . . . . .	lex(1)
update. lsearch,	lfind: linear search and . . . . .	lsearch(3C)
comparison intrinsic/	lge, lgt, lle, llt: string . . . . .	strcmp(3F)
comparison intrinsic/ lge,	lgt, lle, llt: string . . . . .	strcmp(3F)
to subroutines and	libraries. /introduction . . . . .	intro(3)
ar5.0: archive	(library) file format. . . . .	ar5.0(4)
relation for an object	library. /find ordering . . . . .	lorder(1)
relation for an object	library. /find ordering . . . . .	lorder5.0(1)

Permuted Index

ar5.0: archive and portable/ ar: archive and ulimit: get and set user an out-going terminal type, modes, speed, and line: read one common object file. linenum: /ldlinit, ldlfitem: manipulate kdlseek, ldnlseek: seek to an/ strip: strip symbol and nt: out selected fields of each send/cancel requests to an LP	library maintainer. . . . . ar5.0(1) library maintainer for . . . . . ar(1) limits. . . . . ulimit(2) line connection. /establish . . . . . dial(3C) line discipline. /set terminal . . . . . getty(1M) line. . . . . line(1) line number entries in a . . . . . linenum(4) line number entries of a/ . . . . . ldread(3X) line number entries of a/ . . . . . ldseek(3X) line number information from . . . . . strip(1) line numbering filter. . . . . nl(1) line of a file. cut: cut . . . . . cut(1) line printer. lp, cancel: . . . . . lp(1) line: read one line. . . . . line(1) linear search and update. . . . . lsearch(3C) line-feeds. . . . . col(1) linenum: line number entries . . . . . linenum(4) lines common to two sorted . . . . . comm(1) lines. . . . . head(1) lines in a file. . . . . uniq(1) lines of one file. /same lines . . . . . paste(1) lines of several files or . . . . . paste(1) link and unlink system calls. . . . . link(1M) link editor for common object . . . . . ld(1) link editor. . . . . ld5.0(1) link editor output. . . . . a.out(4) link editor output. . . . . a.out5.0(4) link: link to a file. . . . . link(2) link or move files. . . . . cp(1) link to a file. . . . . link(2) link, unlink: exercise link . . . . . link(1M) lint: a C program checker. . . . . lint(1) list contents of directory. . . . . ls(1) list file names and statistics . . . . . ff(1M) list. . . . . nlist(3C) list. . . . . nm5.0(1) list of common object file. . . . . nm(1) list of file systems processed . . . . . checklist(4) list. varargs: . . . . . varargs(5) list. /print formatted . . . . . vprintf(3S) list. /print formatted . . . . . vprintf(3X) listen for connections on a . . . . . listen(2N) listen: listen for connections . . . . . listen(2N) list(s) and execute command. . . . . xargs(1) lle, llt: string comparison . . . . . strcmp(3F) llt: string comparison . . . . . strcmp(3F) ln, mv: copy, link or move . . . . . cp(1) lo: software loopback network . . . . . lo(5) load average statistics. . . . . lav(1) localtime, gmtime, asctime, . . . . . ctime(3C) locate source, binary, and/or . . . . . whereis(1) location of Fortran substring. . . . . index(3F) locations in program. . . . . end(3C) lock process, text, or data in . . . . . plock(2) lockf: record locking on . . . . . lockf(3C) locking on files. . . . . lockf(3C) locking: provide exclusive . . . . . locking(2) log, alog, dlog, clog: Fortran . . . . . log(3F) log gamma function. . . . . gamma(3M) log in to a new group. . . . . newgrp(1) log, log10, pow, sqrt: . . . . . exp(3M) log10, alog10, dlog10: Fortran . . . . . log10(3F) log10, pow, sqrt: exponential, . . . . . exp(3M) logarithm intrinsic function. . . . . log10(3F)
---	---



/dlog, clog: Fortran natural	logarithm intrinsic function. . . . .	log(3F)
/log10, pow, sqrt: exponential,	logarithm, power, square root/	exp(3M)
errpt: process a report of	logged errors. . . . .	errpt(1M)
rwho: who's	logged in on local machines. . . . .	rwho(1N)
getlogin: get	login name. . . . .	getlogin(3C)
logname: get	login name. . . . .	logname(1)
cuserid: get character	login name of the user. . . . .	cuserid(3S)
logname: return	login name of user. . . . .	logname(3X)
passwd: change	login password. . . . .	passwd(1)
rlogin: remote	login. . . . .	rlogin(1N)
rlogind: remote	login server. . . . .	rlogind(8N)
login: sign on. . . . .	login time. profile: . . . . .	login(1)
setting up an environment at	logins of users and teletypes. . . . .	profile(4)
last: indicate last	logname: get login name. . . . .	last(1)
user.	logname: return login name of . . . . .	logname(1)
a64l, l64a: convert between	long integer and base-64 ASCII/ . . . . .	logname(3X)
sputl, sgetl: access	long integer data in a machine/ . . . . .	a64l(3C)
between 3-byte integers and	long integers. /l3tol3: convert . . . . .	sputl(3X)
setjmp,	longjmp: non-local goto. . . . .	l3tol(3C)
lo: software	loopback network interface. . . . .	setjmp(3C)
for an object library.	lord: find ordering relation . . . . .	lo(5)
relation for an object/	lord: find ordering relation . . . . .	lord(1)
mklost+found: make a	lost+found directory for fsck. . . . .	lord5.0(1)
nice: run a command at	low priority. . . . .	mklost+found(1M)
requests to an LP line/	lp, cancel: send/cancel . . . . .	nice(1)
send/cancel requests to an	LP line printer. lp, cancel: . . . . .	lp(1)
disable: enable/disable	LP printers. enable. . . . .	lp(1)
/lpshut, lpmove: start/stop the	LP request scheduler and move/ . . . . .	enable(1)
accept, reject: allow/prevent	LP requests. . . . .	lp sched(1M)
lpadmin: configure the	LP spooling system. . . . .	accept(1M)
lpstat: print	LP status information. . . . .	lpadmin(1M)
spooling system.	lpadmin: configure the LP . . . . .	lpstat(1)
request/ lp sched, lpshut,	lpmove: start/stop the LP . . . . .	lpadmin(1M)
start/stop the LP request/	lp sched, lpshut, lpmove: . . . . .	lp sched(1M)
LP request scheduler/ lp sched,	lpshut, lpmove: start/stop the . . . . .	lp sched(1M)
information.	lpstat: print LP status . . . . .	lpstat(1)
lrand48, / drand48, erand48,	lrand48, rrand48, mrand48, . . . . .	lrand48(3C)
directory.	ls: list contents of . . . . .	ls(1)
and update.	lsearch, lfind: linear search . . . . .	lsearch(3C)
pointer.	lseek: move read/write file . . . . .	lseek(2)
bitwise/ and, or, xor, not,	lshift, rshift: Fortran . . . . .	boof(3F)
integers and long/ l3tol,	l3tol3: convert between 3-byte . . . . .	l3tol(3C)
fscv: convert files between	m4: macro processor. . . . .	m4(1)
provide truth value about/	M68000 and VAX-11/780/ . . . . .	fscv(1M)
/access long integer data in a	m68k, pdp11, u3b, u3b5, vax: . . . . .	machid(1)
put: puts a file onto a remote	machine independent fashion. . . . .	sputl(3X)
takes a file from a remote	machine. . . . .	put(1C)
values:	machine. take: . . . . .	take(1C)
show host status of local	machine-dependent values. . . . .	values(5)
rwho: who's logged in on local	machines. ruptime: . . . . .	ruptime(1N)
update files between two	machines. . . . .	rwho(1N)
update files between two	machines. updater: . . . . .	updater(1)
permutated index. mptx: the	machines. updater: . . . . .	updater(1M)
documents. mm: the MM	macro package for formatting a . . . . .	mptx(5)
mosd: the OSDD adapter	macro package for formatting . . . . .	mm(5)
view graphs and/ mv: a troff	macro package for formatting/ . . . . .	mosd(5)
m4:	macro package for typesetting . . . . .	mv(5)
in this manual. man:	macro processor. . . . .	m4(1)
formatted with the MM	macros for formatting entries . . . . .	man(5)
send mail to users or read	macros. /print/check documents . . . . .	mm(1)
users or read mail.	mail. mail, rmail: . . . . .	mail(1)
netmail: the B-NET network	mail, rmail: send mail to . . . . .	mail(1)
	mail system. . . . .	netmail(8N)

delivermail: deliver	mail to arbitrary people. . . . .	delivermail(8N)
netmailer: deliver	mail to B-NET. . . . .	netmailer(8N)
mail, rmail: send	mail to users or read mail. . . . .	mail(1)
processing system.	mailx: interactive message . . . . .	mailx(1)
malloc, free, realloc, calloc:	main memory allocator. . . . .	malloc(3C)
/mallopt, mallinfo: fast	main memory allocator. . . . .	malloc(3X)
program. ctags:	maintain a tags file for a C . . . . .	ctags(1)
regenerate groups of/ make:	maintain, update, and . . . . .	make(1)
ar5.0: archive and library	maintainer. . . . .	ar5.0(1)
ar: archive and library	maintainer for portable/ . . . . .	ar(1)
intro: introduction to system	maintenance commands and/ . . . . .	intro(1M)
intro: introduction to system	maintenance procedures. . . . .	intro(8)
SCCS file. delta:	make a delta (change) to an . . . . .	delta(1)
mkdir:	make a directory. . . . .	mkdir(1)
or ordinary file. mknod:	make a directory, or a special . . . . .	mknod(2)
for fsck. mklost+found:	make a lost+found directory . . . . .	mklost+fnd(1M)
mktemp:	make a unique filename. . . . .	mktemp(3C)
regenerate groups of/	make: maintain, update, and . . . . .	make(1)
ssp:	make output single spaced. . . . .	ssp(1)
banner:	make posters. . . . .	banner(1)
session. script:	make typescript of terminal . . . . .	script(1)
key.	makekey: generate encryption . . . . .	makekey(1)
/realloc, calloc, mallopt,	mallinfo: fast main memory/ . . . . .	malloc(3X)
main memory allocator.	malloc, free, realloc, calloc: . . . . .	malloc(3C)
mallopt, mallinfo: fast main/	malloc, free, realloc, calloc, . . . . .	malloc(3X)
malloc, free, realloc, calloc,	mallopt, mallinfo: fast main/ . . . . .	malloc(3X)
entries in this manual.	man: macros for formatting . . . . .	man(5)
manual.	man: print entries in this . . . . .	man(1)
/tfind, tdelete, twalk:	manage binary search trees. . . . .	tsearch(3C)
hsearch, hcreate, hdestroy:	manage hash search tables. . . . .	hsearch(3C)
sh: shell layer	manager. . . . .	sh(1)
records. fwtmp, wtmpfix:	manipulate connect accounting . . . . .	fwtmp(1M)
of/ ldread, ldlini, ldliem:	manipulate line number entries . . . . .	ldread(3X)
frexp, ldexp, modf:	manipulate parts of/ . . . . .	frexp(3C)
tp:	manipulate tape archive. . . . .	tp(1)
route: manually	manipulate the routing tables. . . . .	route(8N)
/inet_netof: Internet address	manipulation routines. . . . .	inet(3N)
locate source, binary, and/or	manual for program. whereis: . . . . .	whereis(1)
man: print entries in this	manual. . . . .	man(1)
for formatting entries in this	manual. man: macros . . . . .	man(5)
routing tables. route:	manually manipulate the . . . . .	route(8N)
ascii:	map of ASCII character set. . . . .	ascii(5)
files. diffmk:	mark differences between . . . . .	diffmk(1)
umask: set file-creation mode	mask. . . . .	umask(1)
set and get file creation	mask. umask: . . . . .	umask(2)
an error message file by	massaging C source. /create . . . . .	mkstr(1)
table. master:	master device information . . . . .	master(4)
information table.	master: master device . . . . .	master(4)
regular expression compile and	match routines. regexp: . . . . .	regexp(5)
math:	math functions and constants. . . . .	math(5)
constants.	math: math functions and . . . . .	math(5)
eqn, neqn, checkq: format	mathematical text for nroff or/ . . . . .	eqn(1)
function.	matherr: error-handling . . . . .	matherr(3M)
dmaxl: Fortran maximum-value/	max, max0, amax0, max1, amax1, . . . . .	max(3F)
dmaxl: Fortran/ max,	max0, amax0, max1, amax1, . . . . .	max(3F)
max, max0, amax0,	max1, amax1, dmaxl: Fortran/ . . . . .	max(3F)
/max1, amax1, dmaxl: Fortran	maximum-value functions. . . . .	max(3F)
	maze: generate a maze. . . . .	maze(6)
maze: generate a	maze. . . . .	maze(6)
	mc68cc: C compiler. . . . .	mc68cc(1)
accounting.	mclock: return Fortran time . . . . .	mclock(3F)
bcd: convert to antique	media. . . . .	bcd(6)
memcpy, memset: memory/	mem, kmem: core memory. . . . .	mem(7)
	memcpy, memchr, memcmp, . . . . .	memory(3C)

**memset:** memory/ memcopy, memchr, memcopy, memchr, memchr, memset: memory(3C)  
**operations.** memcopy, memchr, memcopy, memchr, memchr, memset: memory(3C)  
**memcopy, memchr, memcmp,** memcopy, memset: memory/ . . . . . memory(3C)  
**free, realloc, calloc:** main  
**malloc, mallinfo:** fast main  
**shmctl:** shared  
**queue, semaphore set or shared**  
**mem, kmem:** core  
**memcmp, memcopy, memset:**  
**shmop:** shared  
**lock process, text, or data in**  
**shmget:** get shared  
**/memchr, memcmp, memcopy,**  
**sort:** sort and/or  
**files.** acctmerg:  
**files or subsequent/ paste:**  
  
**msgctl:**  
**mkstr:** create an error  
**recvfrom, recvmsg:** receive a  
**send, sendto, sendmsg:** send a  
**msgop:**  
**mailx:** interactive  
**msgget:** get  
**or shared/ ipcrm:** remove a  
**msg:** permit or deny  
**sys\_err:** system error  
**dmin1:** Fortran minimum-value/  
**dmin1:** Fortran/ min,  
**min, min0, amin0,**  
**/min1, amin1, dmin1:** Fortran  
  
**system.**  
**lost+found directory for/**  
**special or ordinary file.**  
**file by massaging C source.**  
**filename.**  
**formatting documents.** mm: the  
**documents formatted with the**  
**documents formatted with the/**  
**formatting documents.**  
**view graphs, and slides.**  
**table.**  
**remaindering intrinsic/**  
**chmod:** change  
**umask:** set file-creation  
**chmod:** change  
**getty:** set terminal type,  
**bs:** a compiler/interpreter for  
**floating-point/ frexp, ldexp,**  
**touch:** update access and  
**utime:** set file access and  
**/ckpact, dodisk, lastlogin,**  
**profile.**  
**uuser:**  
  
**package for formatting/**  
**rcvhex:** translates  
**mount:**  
**system.** mount, umount:  
  
**setmnt:** establish  
**memchr, memcmp, memcopy, . . . . . memory(3C)**  
**memcmp, memcopy, memset: memory** memory(3C)  
**memcopy, memset: memory/ . . . . . memory(3C)**  
**memory allocator. malloc, . . . . . malloc(3C)**  
**memory allocator. /calloc, . . . . . malloc(3X)**  
**memory control operations. . . . . shmctl(2)**  
**memory id. /remove a message . . . . . ipcrm(1)**  
**memory. . . . . mem(7)**  
**memory operations. /memchr, . . . . . memory(3C)**  
**memory operations. . . . . shmop(2)**  
**memory. plock: . . . . . plock(2)**  
**memory segment. . . . . shmget(2)**  
**memset: memory operations. . . . . memory(3C)**  
**merge files. . . . . sort(1)**  
**merge or add total accounting . . . . . acctmerg(1M)**  
**merge same lines of several . . . . . paste(1)**  
**msg: permit or deny messages. . . . . msg(1)**  
**message control operations. . . . . msgctl(2)**  
**message file by massaging C/ . . . . . mkstr(1)**  
**message from a socket. recv, . . . . . recv(2N)**  
**message from a socket. . . . . send(2N)**  
**message operations. . . . . msgop(2)**  
**message processing system. . . . . mailx(1)**  
**message queue. . . . . msgget(2)**  
**message queue, semaphore set . . . . . ipcrm(1)**  
**messages. . . . . msg(1)**  
**messages. /errno, sys\_errlist, . . . . . perror(3C)**  
**min, min0, amin0, min1, amin1, . . . . . min(3F)**  
**min0, amin0, min1, amin1, . . . . . min(3F)**  
**min1, amin1, dmin1:** Fortran/  
**minimum-value functions. . . . . min(3F)**  
**mkdir:** make a directory. . . . . mkdir(1)  
**mkfs:** construct a file system. . . . . mkfs(1M)  
**mkfs1b:** construct a file . . . . . mkfs1b(1M)  
**mklost+found:** make a . . . . . mklost+found(1M)  
**mknod:** build special file. . . . . mknod(1M)  
**mknod:** make a directory, or a . . . . . mknod(2)  
**mkstr:** create an error message . . . . . mkstr(1)  
**mktemp:** make a unique . . . . . mktemp(3C)  
**MM macro package for . . . . . mm(5)**  
**MM macros. /print/check . . . . . mm(1)**  
**mm, osdd, checkmm: print/check . . . . . mm(1)**  
**mm: the MM macro package for . . . . . mm(5)**  
**mmt, mvt:** typeset documents, . . . . . mmt(1)  
**mnttab:** mounted file system . . . . . mnttab(4)  
**mod, amod, dmod:** Fortran . . . . . mod(3F)  
**mode. . . . . chmod(1)**  
**mode mask. . . . . umask(1)**  
**mode of file. . . . . chmod(2)**  
**modes, speed, and line/ . . . . . getty(1M)**  
**modest-sized programs. . . . . bs(1)**  
**modf:** manipulate parts of . . . . . frexp(3C)  
**modification times of a file. . . . . touch(1)**  
**modification times. . . . . utime(2)**  
**monacct, nulladm, prttmp,/ . . . . . acctsh(1M)**  
**monitor: prepare execution . . . . . monitor(3C)**  
**monitor uucp network. . . . . uuser(1M)**  
**moo: guessing game. . . . . moo(6)**  
**mosd: the OSDD adapter macro . . . . . mosd(5)**  
**Motorola S-records from/ . . . . . rcvhex(1)**  
**mount a file system. . . . . mount(2)**  
**mount and dismount file . . . . . mount(1M)**  
**mount: mount a file system. . . . . mount(2)**  
**mount table. . . . . setmnt(1M)**

*Permuted Index*

dismount file system.	mount, umount: mount and	mount(1M)
mnttab:	mounted file system table.	mnttab(4)
mvdir:	move a directory.	mvdir(1M)
cp, ln, mv: copy, link or	move files.	cp(1)
lseek:	move read/write file pointer.	lseek(2)
the LP request scheduler and	move requests. /start/stop	lpsched(1M)
formatting a permuted index.	mptx: the macro package for	mptx(5)
/rand48, lrand48, nrand48,	mrnd48, jrnd48, srnd48./	drand48(3C)
operations.	msgctl: message control	msgctl(2)
	msgget: get message queue.	msgget(2)
	msgop: message operations.	msgop(2)
select: synchronous i/o	multiplexing.	select(2N)
typesetting view graphs and/	mv: a troff macro package for	mv(5)
cp, ln,	mv: copy, link or move files.	cp(1)
	mvdir: move a directory.	mvdir(1M)
graphs, and slides. mmt,	mvt: typeset documents, view	mmt(1)
log, alog, dlog, clog: Fortran	natural logarithm intrinsic/	log(3F)
i-numbers.	ncheck: generate names from	ncheck(1M)
/dnint, nint, idnint: Fortran	nearest integer functions.	round(3F)
mathematical text for/ eqn,	neqn, checkeq: format	eqn(1)
definitions for eqn and	neqn. /special character	eqnchar(5)
mail system.	netmail: the B-NET network	netmail(8N)
B-NET.	netmailer: deliver mail to	netmailer(8N)
	netstat: show network status.	netstat(1N)
values between host and	network byte order. /convert	byteorder(3N)
setnetent, endnetent: get	network entry. /getnetbyname,	getnetent(3N)
/sethostent, endhostent: get	network host entry.	gethostent(3N)
lo: software loopback	network interface.	lo(5)
ifconfig: configure	network interface parameters.	ifconfig(8N)
netmail: the B-NET	network mail system.	netmail(8N)
networks:	network name data base.	networks(4N)
routed:	network routing daemon.	routed(8N)
netstat: show	network status.	netstat(1N)
uusub: monitor uucp	network.	uusub(1M)
networking: introduction to	networking facilities.	intro(5N)
networking facilities.	networking: introduction to	intro(5N)
base.	networks: network name data	networks(4N)
a text file.	newform: change the format of	newform(1)
	newgrp: log in to a new group.	newgrp(1)
news: print	news items.	news(1)
	news: print news items.	news(1)
process.	nice: change priority of a	nice(2)
priority.	nice: run a command at low	nice(1)
integer/ anint, dnint,	nint, idnint: Fortran nearest	round(3F)
	nl: line numbering filter.	nl(1)
list.	nlist: get entries from name	nlist(3C)
object file.	nm: print name list of common	nm(1)
	nm5.0: print name list.	nm5.0(1)
change current UNIX system	nodename. chgnod:	chgnod(1M)
hangups (sh only).	nohup: run a command immune to	nohup(1)
setjmp, longjmp:	non-local goto.	setjmp(3C)
bitwise boolean/ and, or, xor,	not, lshift, rshift: Fortran	bool(3F)
drand48, erand48, lrand48,	nrand48, mrnd48, jrnd48./	drand48(3C)
	nroff: format text.	nroff(1)
format mathematical text for	nroff or troff. /checkeq:	eqn(1)
tbl: format tables for	nroff or troff.	tbl(1)
constructs. deroff: remove	nroff/troff, tbl, and eqn	deroff(1)
between host/ htont, htons,	ntohl, ntohs: convert values	byteorder(3N)
host and/ htont, htons, ntohl,	ntohs: convert values between	byteorder(3N)
null: the	null file.	null(7)
	null: the null file.	null(7)
/dodisk, lastlogin, monacct,	nulladm, prctmp, prdaily./	acctsh(1M)
nl: line	numbering filter.	nl(1)
number: convert Arabic	numerals to English.	number(6)

ldfcn: common	object file access routines. . . . .	ldfcn(4)
conv: object file converter. . . . .		conv(1)
dump selected parts of an	object file. dump: . . . . .	dump(1)
ldopen, ldaopen: open a common	object file for reading. . . . .	ldopen(3X)
number entries of a common	object file function. /line . . . . .	ldread(3X)
ldaclose: close a common	object file. kclose, . . . . .	kclose(3X)
the file header of a common	object file. ldhread: read . . . . .	ldhread(3X)
retrieve symbol name for	object file. ldgetname: . . . . .	ldgetname(3X)
of a section of a common	object file. /number entries . . . . .	ldlseek(3X)
file header of a common	object file. /to the optional . . . . .	ldohseek(3X)
of a section of a common	object file. /entries . . . . .	ldrseek(3X)
section header of a common	object file. /an indexed/named . . . . .	ldshread(3X)
section of a common	object file. /an indexed/named . . . . .	ldsseek(3X)
symbol table entry of a common	object file. /the index of a . . . . .	ldtbindex(3X)
symbol table entry of a common	object file. /read an indexed . . . . .	ldtbread(3X)
the symbol table of a common	object file. /seek to . . . . .	ldtbseek(3X)
number entries in a common	object file. linenum: line . . . . .	linenum(4)
nm: print name list of common	object file. . . . .	nm(1)
information for a common	object file. /relocation . . . . .	reloc(4)
section header for a common	object file. scnhdr: . . . . .	scnhdr(4)
size5.0: size of an	object file. . . . .	size5.0(1)
number information from an	object file. /symbol and line . . . . .	strip(1)
format. syms: common	object file symbol table . . . . .	syms(4)
- a.out header for common	object files. aouthdr.h . . . . .	aouthdr(4)
file header for common	object files. filehdr: . . . . .	filehdr(4)
hex: translates	object files. . . . .	hex(1)
directories. cpset: install	object files in binary . . . . .	cpset(1M)
ld: link editor for common	object files. . . . .	ld(1)
print section sizes of common	object files. size: . . . . .	size(1)
find ordering relation for an	object library. lorder: . . . . .	lorder(1)
find ordering relation for an	object library. lorder5.0: . . . . .	lorder5.0(1)
/the printable strings in an	object, or other binary file. . . . .	strings(1)
/setgrent, endgrent, fgetgrent:	obtain group file entry from <i>u</i> / . . . . .	getgrent(3C)
od:	octal dump. . . . .	od(1)
command immune to hangups (sh	od: octal dump. . . . .	od(1)
the specified/ exterr: turn	only. nohup: run a . . . . .	nohup(1)
put: puts a file	on/off the extended errors in . . . . .	exterr(1)
reading. ldopen, ldaopen:	onto a remote machine. . . . .	put(1C)
fopen, freopen, fdopen:	open a common object file for . . . . .	ldopen(3X)
open:	open a stream. . . . .	fopen(3S)
writing.	open for reading or writing. . . . .	open(2)
seekdir, rewinddir, closedir:/	open: open for reading or . . . . .	open(2)
tputs: terminal independent	opendir, readdir, telldir, . . . . .	directory(3X)
ffs: bit and byte string	operation routines. /goto, . . . . .	termcap(3X)
flexible length directory	operations. /bcmp, bzero, . . . . .	bstring(3N)
memcmp, memcmp, memset: memory	operations. /closedir: . . . . .	directory(3X)
msgctl: message control	operations. memccpy, memchr, . . . . .	memory(3C)
msgop: message	operations. . . . .	msgctl(2)
semctl: semaphore control	operations. . . . .	msgop(2)
semop: semaphore	operations. . . . .	semctl(2)
shmctl: shared memory control	operations. . . . .	semop(2)
shmop: shared memory	operations. . . . .	shmctl(2)
strncpy, strtok: string	operations. /strpbrk, strspn, . . . . .	shmop(2)
join: relational database	operator. . . . .	string(3C)
dcopy: copy file systems for	optimal access time. . . . .	join(1)
CRT screen handling and	optimization package. curses: . . . . .	dcopy(1M)
vector. getopt: get	option letter from argument . . . . .	curses(3X)
common/ ldohseek: seek to the	optional file header of a . . . . .	getopt(3C)
fcntl: file control	options. . . . .	ldohseek(3X)
stty: set the	options for a terminal. . . . .	fcntl(5)
getopt: parse command	options. . . . .	stty(1)
/setsockopt: get and set	options on sockets. . . . .	getopt(1)
Fortran bitwise boolean/ and,	or, xor, not, lshift, rshift: . . . . .	setsockopt(2N)
		bool(3F)

Permuted Index

object library. lorder: find  
   object/ lorder5.0: find  
 a directory, or a special or  
   formatting/ mosd: the  
 documents formatted with/ mm,  
   dial: establish an  
   assembler and link editor  
   assembler and link editor  
   /vsprintf: print formatted  
   /vsprintf: print formatted  
   sprintf: print formatted  
     ssp: make  
 /acctdusg, accton, acctwtmp:  
   chown: change  
     chown, chgrp: change  
       and expand files.  
   handling and optimization  
   permuted/ mptx: the macro  
 documents. mm: the MM macro  
 mosd: the OSDD adapter macro  
   graphs and/ mv: a troff macro  
   sadc: system activity report  
   standard buffered input/output  
   interprocess communication  
   4014 terminal. 4014:  
 tune floppy disk settling time  
   configure network interface  
   process, process group, and  
   getopt:  
  
   /endpwent, fgetpwent: get  
     putpwent: write  
       passwd:  
         getpass: read a  
         passwd: change login  
           pwck, grpck:  
 several files or subsequent/  
 dirname: deliver portions of  
   directory. getcwd: get  
   fgrep: search a file for a  
   processing language. awk:  
     signal.  
     expand files. pack,  
     a process. popen,  
 truth value about your/ m68k,  
   get name of connected  
   messg:  
 macro package for formatting a  
   ptx:  
     format. acct:  
 acctcms: command summary from  
   sys\_err: system error/  
   viewing. more: file  
   terminals. pg: file  
   soft-copy terminals.  
   tc:  
 access physical addresses.  
   allow a process to access  
   split: split a file into  
     channel.  
     tee:  
   popen, pclose: initiate  
   fish:  
 ordering relation for an . . . . . lorder(1)  
 ordering relation for an . . . . . lorder5.0(1)  
 ordinary file. mknod: make . . . . . mknod(2)  
 OSDD adapter macro package for . . . . . mosd(5)  
 osdd, checkmm: print/check . . . . . mm(1)  
 out-going terminal line/ . . . . . dial(3C)  
 output. a.out: common . . . . . a.out(4)  
 output. a.out5.0: . . . . . a.out5.0(4)  
 output of a varargs argument/ . . . . . vprintf(3S)  
 output of a varargs argument/ . . . . . vprintf(3X)  
 output. printf, sprintf, . . . . . printf(3S)  
 output single spaced. . . . . ssp(1)  
 overview of accounting and/ . . . . . acct(1M)  
 owner and group of a file. . . . . chown(2)  
 owner or group. . . . . chown(1)  
 pack, pcat, unpack: compress . . . . . pack(1)  
 package. curses: CRT screen . . . . . curses(3X)  
 package for formatting a . . . . . mptx(5)  
 package for formatting . . . . . mm(5)  
 package for formatting/ . . . . . mosd(5)  
 package for typesetting view . . . . . mv(5)  
 package. sai, sa2, . . . . . sar(1M)  
 package. stdio: . . . . . stdio(3S)  
 package. ftok: standard . . . . . stdipc(3C)  
 paginator for the Tektronix . . . . . 4014(1)  
 parameters. disk tune: . . . . . disk tune(1M)  
 parameters. ifconfig: . . . . . ifconfig(8N)  
 parent process IDs. /get . . . . . getpid(2)  
 parse command options. . . . . getopt(1)  
 passwd: change login password. . . . . passwd(1)  
 passwd: password file. . . . . passwd(4)  
 password file entry. . . . . getpwent(3C)  
 password file entry. . . . . putpwent(3C)  
 password file. . . . . passwd(4)  
 password. . . . . getpass(3C)  
 password. . . . . passwd(1)  
 password/group file checkers. . . . . pwck(1M)  
 paste: merge same lines of . . . . . paste(1)  
 path names. basename, . . . . . basename(1)  
 pathname of current working . . . . . getcwd(3C)  
 pattern. grep, egrep, . . . . . grep(1)  
 pattern scanning and . . . . . awk(1)  
 pause: suspend process until . . . . . pause(2)  
 pcat, unpack: compress and . . . . . pack(1)  
 pclose: initiate pipe to/from . . . . . popen(3S)  
 pdp11, u3b, u3b5, vax: provide . . . . . machid(1)  
 peer. getpeername: . . . . . getpeername(2N)  
 permit or deny messages. . . . . mesg(1)  
 permuted index. mptx: the . . . . . mptx(5)  
 permuted index. . . . . ptx(1)  
 per-process accounting file . . . . . acct(4)  
 per-process accounting/ . . . . . acctcms(1M)  
 perror, errno, sys\_errlist, . . . . . perror(3C)  
 perusal filter for crt . . . . . more(1)  
 perusal filter for soft-copy . . . . . pg(1)  
 pg: file perusal filter for . . . . . pg(1)  
 phototypesetter simulator. . . . . tc(1)  
 phys: allow a process to . . . . . phys(2)  
 physical addresses. phys: . . . . . phys(2)  
 pieces. . . . . split(1)  
 pipe: create an interprocess . . . . . pipe(2)  
 pipe fitting. . . . . tee(1)  
 pipe to/from a process. . . . . popen(3S)  
 play "Go Fish". . . . . fish(6)

life:	play the game of life. . . . .	life(6)
worm:	Play the growing worm game. . . . .	worm(6)
data in memory.	plock: lock process, text, or	plock(2)
	plot: graphics interface. . . . .	plot(4)
subroutines.	plot: graphics interface. . . . .	plot(3X)
images.	pnch: file format for card	pnch(4)
ftell: reposition a file	pointer in a stream. /rewind,	fseek(3S)
lseek: move read/write file	pointer. . . . .	lseek(2)
to/from a process.	popen, pclose: initiate pipe	popen(3S)
data base of terminal types by	port. ttytype: . . . . .	ttytype(4)
and library maintainer for	portable archives. /archive	ar(1)
basename, dirname: deliver	portions of path names. . . . .	basename(1)
functions. dim, ddim, idim:	positive difference intrinsic	dim(3F)
banner: make	posters. . . . .	banner(1)
logarithm./ exp, log, log10,	pow, sqrt: exponential,	exp(3M)
/sqrt: exponential, logarithm,	power, square root functions.	exp(3M)
brc, bcheckrc, rc,	powerfail: system/ . . . . .	brc(1M)
/lastlogin, monacct, nulladm,	pr: print files. . . . .	pr(1)
/monacct, nulladm, prttmp,	prttmp, prdaily, prtacct,/	acctsh(1M)
function. dprod: double	prdaily, prtacct, runacct,/	acctsh(1M)
for troff. cw, checkcw:	precision product intrinsic	dprod(3F)
monitor:	prepare constant-width text	cw(1)
cpp: the C language	prepare execution profile. . . . .	monitor(3C)
cpp: the C language	preprocessor. . . . .	cpp(1)
unset: undo a	preprocessor. . . . .	cpp5.0(1)
types:	previous get of an SCCS file.	unset(1)
interesting, adage. fortune:	primitive system data types.	types(5)
pr:	print a random, hopefully	fortune(6)
date:	print an SCCS file. . . . .	prs(1)
cal:	print and set the date. . . . .	date(1)
of a file. sum:	print calendar. . . . .	cal(1)
editing activity. sact:	print checksum and block count	sum(1)
id. whoami:	print current SCCS file . . . . .	sact(1)
man:	print effective current user	whoami(1)
cat: concatenate and	print entries in this manual. . . . .	man(1)
pr:	print files. . . . .	cat(1)
vprintf, vfprintf, vsprintf:	print files. . . . .	pr(1)
vprintf, vfprintf, vsprintf:	print formatted output of a/	vprintf(3S)
printf, fprintf, sprintf:	print formatted output of a/	vprintf(3X)
host system. hostid: set or	print formatted output. . . . .	printf(3S)
banner7:	print identifier of current	hostid(1N)
lav:	print large banner on printer.	banner7(1)
lpstat:	print load average statistics. . . . .	lav(1)
nm5.0:	print LP status information. . . . .	lpstat(1)
object file. nm:	print name list. . . . .	nm5.0(1)
system. hostname: set or	print name list of common	nm(1)
system. uname:	print name of current host	hostname(1N)
news:	print name of current UNIX	uname(1)
printenv:	print news items. . . . .	news(1)
file(s). acctcom: search and	print out the environment.	printenv(1)
object files. size:	print process accounting	acctcom(1)
pstat:	print section sizes of common	size(1)
names. id:	print system facts. . . . .	pstat(1M)
formatted/ mm, osdd, checkmm:	print user and group IDs and	id(1)
environment.	printable strings in an	strings(1)
banner7: print large banner on	print/check documents . . . . .	mm(1)
requests to an LP line	printenv: print out the	printenv(1)
disable: enable/disable LP	printer. . . . .	banner7(1)
print formatted output.	printer. /cancel: send/cancel	lp(1)
nice: run a command at low	printers. enable, . . . . .	enable(1)
nice: change	printf, fprintf, sprintf: . . . . .	printf(3S)
errors. errpt:	priority. . . . .	nice(1)
	priority of a process. . . . .	nice(2)
	process a report of logged	errpt(1M)

*Permuted Index*

acct: enable or disable	process accounting. . . . .	acct(2)
acctprcl, acctprc2:	process accounting. . . . .	acctprc(1M)
acctcom: search and print	process accounting file(s). . . . .	acctcom(1)
times. times: get	process and child process . . . . .	times(2)
init, telinit:	process control/ . . . . .	init(1M)
timex: time a command; report	process data and system/ . . . . .	timex(1)
exit, _exit: terminate	process. . . . .	exit(2)
fork: create a new	process. . . . .	fork(2)
/getpgrp, getppid: get process,	process group, and parent/ . . . . .	getpid(2)
setpgrp: set	process group ID. . . . .	setpgrp(2)
killpg: send signal to a	process group. . . . .	killpg(3N)
process group, and parent	process IDs. /get process, . . . . .	getpid(2)
inittab: script for the init	process. . . . .	inittab(4)
kill: terminate a	process. . . . .	kill(1)
nice: change priority of a	process. . . . .	nice(2)
kill: send a signal to a	process or a group of/ . . . . .	kill(2)
initiate pipe to/from a	process. popen, pclose: . . . . .	popen(3S)
getpid, getpgrp, getppid: get	process, process group, and/ . . . . .	getpid(2)
ps: report	process status. . . . .	ps(1)
memory. plock: lock	process, text, or data in . . . . .	plock(2)
times: get process and child	process times. . . . .	times(2)
addresses. phys: allow a	process to access physical . . . . .	phys(2)
wait: wait for child	process to stop or terminate. . . . .	wait(2)
wait3: wait for child	process to stop or terminate. . . . .	wait3(2N)
ptrace:	process trace. . . . .	ptrace(2)
pause: suspend	process until signal. . . . .	pause(2)
list of file systems	processed by fsck. checklist: . . . . .	checklist(4)
to a process or a group of	processes. /send a signal . . . . .	kill(2)
killall: kill all active	processes. . . . .	killall(1M)
structure. fuser: identify	processes using a file or file . . . . .	fuser(1M)
awk: pattern scanning and	processing language. . . . .	awk(1)
shutdown: terminate all	processing. . . . .	shutdown(1M)
mailx: interactive message	processing system. . . . .	mailx(1)
m4: macro	processor. . . . .	m4(1)
provide truth value about your	processor type. /u3b5, vax: . . . . .	machid(1)
between M68000 and VAX-11/780	processors. /convert files . . . . .	fcv(1M)
alarm: set a	process's alarm clock. . . . .	alarm(2)
dprod: double precision	product intrinsic function. . . . .	dprod(3F)
	prof: display profile data. . . . .	prof(1)
	prof: profile within a . . . . .	prof(5)
	profit: execution time . . . . .	profl(2)
	profile data. . . . .	prof(1)
	profile. . . . .	monitor(3C)
	profile. . . . .	profl(2)
monitor: prepare execution	profile: setting up an . . . . .	profile(4)
profil: execution time	profile within a function. . . . .	prof(5)
environment at login time.	programming language. /the . . . . .	sh(1)
prof:	Protocol. . . . .	arp(5P)
standard/restricted command	protocol entry. . . . .	getprotoent(3N)
arp: Address Resolution	protocol family. . . . .	inet(5F)
/setprotoent, endprotoent: get	Protocol. . . . .	ip(5P)
inet: Internet	protocol name data base. . . . .	protocols(4N)
ip: Internet	Protocol server. ftpd: . . . . .	ftpd(8N)
protocols:	protocol server. . . . .	telnetd(8N)
DARPA Internet File Transfer	Protocol server. tftpd: . . . . .	tftpd(8N)
telnetd: DARPA TELNET	Protocol. tcp: . . . . .	tcp(5P)
DARPA Trivial File Transfer	protocol. telnet: . . . . .	telnet(1N)
Internet Transmission Control	protocol trace. . . . .	trpt(8N)
user interface to the TELNET	Protocol. . . . .	udp(5P)
trpt: transliterate	protocols: protocol name data . . . . .	protocols(4N)
udp: Internet User Datagram	provide drill in number facts. . . . .	arithmetic(6)
base.	provide exclusive file regions . . . . .	locking(2)
arithmeti:	provide truth value about your/ . . . . .	machid(1)
for reading or/ locking:	provide truth values. . . . .	true(1)
m68k, pdp11, u3b, u3b5, vax:		
true, false:		



/nulladm, pretmp, prdaily,	prs: print an SCCS file. . . . .	prs(1)
	prtacct, runacct, shutacct,/ . . . . .	acctsh(1M)
	ps: report process status. . . . .	ps(1)
	pty: pseudo terminal driver. . . . .	pty(5)
	sxt: pseudo-device driver. . . . .	sxt(7)
/generate uniformly distributed	pseudo-random numbers. . . . .	drand48(3C)
	pstat: print system facts. . . . .	pstat(1M)
	ptrace: process trace. . . . .	ptrace(2)
	ptx: permuted index. . . . .	ptx(1)
	pty: pseudo terminal driver. . . . .	pty(5)
stream. ungetc:	push character back into input . . . . .	ungetc(3S)
put character or word on a/	putc, putchar, fputc, putw: . . . . .	putc(3S)
character or word on a/	putchar, fputc, putw: put . . . . .	putc(3S)
environment.	putenv: change or add value to . . . . .	putenv(3C)
entry.	putpwent: write password file . . . . .	putpwent(3C)
machine. put:	puts a file onto a remote . . . . .	put(1C)
stream.	puts, fputs: put a string on a . . . . .	puts(3S)
getutent, getutid, getutline,	pututline, setutent, endutent,/ . . . . .	getut(3C)
a/ putc, putchar, fputc,	putw: put character or word on . . . . .	putc(3S)
file checkers.	pwck, grpck: password/group . . . . .	pwck(1M)
	pwd: working directory name. . . . .	pwd(1)
	qsort: quicker sort. . . . .	qsort(3C)
tput:	query terminfo database. . . . .	tput(1)
insert/remove element from a	queue. insque, remque: . . . . .	insque(3N)
msgget: get message	queue. . . . .	msgget(2)
ipcrm: remove a message	queue, semaphore set or shared/ . . . . .	ipcrm(1)
qsort:	quicker sort. . . . .	qsort(3C)
	quiz: test your knowledge. . . . .	quiz(6)
display.	rain: animated raindrops . . . . .	rain(6)
rain: animated	raindrops display. . . . .	rain(6)
random-number/ irand, srand,	rand: Fortran uniform . . . . .	rand(3F)
random-number generator.	rand, srand: simple . . . . .	rand(3C)
adage. fortune: print a	random, hopefully interesting, . . . . .	fortune(6)
rand, srand: simple	random-number generator. . . . .	rand(3C)
/srand, rand: Fortran uniform	random-number generator. . . . .	rand(3F)
fsplit: split f77,	ratfor, or efi files. . . . .	fsplit(1)
dialect.	ratfor: rational Fortran . . . . .	ratfor(1)
ratfor:	rational Fortran dialect. . . . .	ratfor(1)
initialization/ brc, bcheckrc,	rc, powerfail: system . . . . .	brc(1M)
routines for returning a/	rcmd, rresvport, ruserok: . . . . .	rcmd(3N)
	rcp: remote file copy. . . . .	rcp(1N)
S-records from downloading/	rcvhex: translates Motorola . . . . .	rcvhex(1)
getpass:	read a password. . . . .	getpass(3C)
entry of a common/ ldtbread:	read an indexed symbol table . . . . .	ldtbread(3X)
header/ ldshread, ldnsbread:	read an indexed/named section . . . . .	ldshread(3X)
read:	read from file. . . . .	read(2)
readv:	read from file. . . . .	readv(3N)
rmail: send mail to users or	read mail. mail, . . . . .	mail(1)
line:	read one line. . . . .	line(1)
	read: read from file. . . . .	read(2)
member of an/ ldahread:	read the archive header of a . . . . .	ldahread(3X)
common object file. ldfhread:	read the file header of a . . . . .	ldfhread(3X)
rewinddir, closedir:/ opendir,	readdir, telldir, seekdir, . . . . .	directory(3X)
open a common object file for	reading. llopen, ldaopen: . . . . .	llopen(3X)
exclusive file regions for	reading or writing. /provide . . . . .	locking(2)
open: open for	reading or writing. . . . .	open(2)
	readv: read from file. . . . .	readv(3N)
lseek: move	read/write file pointer. . . . .	lseek(2)
emplx,/ int, ifix, idint,	real, float, snl, dbl, . . . . .	ftype(3F)
allocator. malloc, free,	realloc, calloc: main memory . . . . .	malloc(3C)
mallinfo: fast/ malloc, free,	realloc, calloc, mallot, . . . . .	malloc(3X)
	reboot: reboot the system. . . . .	reboot(1M)
	reboot: reboot the system. . . . .	reboot(2)
reboot:	reboot the system. . . . .	reboot(1M)

*Permuted Index*

reboot:	reboot the system. . . . .	reboot(2)
specify what to do upon	receipt of a signal. signal: . . . . .	signal(2)
/specify Fortran action on	receipt of a system signal. . . . .	signal(3F)
recv, recvfrom, recvmsg:	receive a message from a/ . . . . .	recv(2N)
lockf:	record locking on files. . . . .	lockf(3C)
from per-process accounting	records. /command summary . . . . .	acctcms(1M)
errdead: extract error	records from dump. . . . .	errdead(1M)
manipulate connect accounting	records. fwtmp, wtmpfix: . . . . .	fwtmp(1M)
tape. freq:	recover files from a backup . . . . .	freq(1M)
receive a message from a/	recv, recvfrom, recvmsg: . . . . .	recv(2N)
message from a socket. recv,	recvfrom, recvmsg: receive a . . . . .	recv(2N)
from a/ recv, recvfrom,	recvmsg: receive a message . . . . .	recv(2N)
ed,	red: text editor. . . . .	ed(1)
execute a regular expression.	regcmp, regex: compile and . . . . .	regcmp(3X)
compile.	regcmp: regular expression . . . . .	regcmp(1)
make: maintain, update, and	regenerate groups of programs. . . . .	make(1)
regular expression. regcmp,	regex: compile and execute a . . . . .	regcmp(3X)
compile and match routines.	regexp: regular expression . . . . .	regexp(5)
/provide exclusive file	regions for reading or/ . . . . .	locking(2)
match routines. regexp:	regular expression compile and . . . . .	regexp(5)
regcmp:	regular expression compile. . . . .	regcmp(1)
regex: compile and execute a	regular expression. regcmp. . . . .	regcmp(3X)
requests. accept,	reject: allow/prevent LP . . . . .	accept(1M)
sorted files. comm: select or	reject lines common to two . . . . .	comm(1)
lorder: find ordering	relation for an object/ . . . . .	lorder(1)
lorder5.0: find ordering	relation for an object/ . . . . .	lorder5.0(1)
join:	relational database operator. . . . .	join(1)
for a common object file.	reloc: relocation information . . . . .	reloc(4)
strip5.0: remove symbols and	relocation bits. . . . .	strip5.0(1)
ldrseek, ldrseek: seek to	relocation entries of a/ . . . . .	ldrseek(3X)
common object file. reloc:	relocation information for a . . . . .	reloc(4)
/fmod, fabs: floor, ceiling,	remainder, absolute value/ . . . . .	floor(3M)
mod, amod, dmod: Fortran	remaindering intrinsic/ . . . . .	mod(3F)
calendar:	reminder service. . . . .	calendar(1)
for returning a stream to a	remote command. /routines . . . . .	rcmd(3N)
rexec: return stream to a	remote command. . . . .	rexec(3N)
rexecd:	remote execution server. . . . .	rexecd(8N)
rcp:	remote file copy. . . . .	rcp(1N)
rlogin:	remote login. . . . .	rlogin(1N)
rlogind:	remote login server. . . . .	rlogind(8N)
put: puts a file onto a	remote machine. . . . .	put(1C)
take: takes a file from a	remote machine. . . . .	take(1C)
remsh:	remote shell. . . . .	remsh(1N)
remshd:	remote shell server. . . . .	remshd(8N)
ct: spawn getty to a	remote terminal. . . . .	ct(1C)
file. rmdel:	remove a delta from an SCCS . . . . .	rmdel(1)
semaphore set or/ ipcrm:	remove a message queue. . . . .	ipcrm(1)
unlink:	remove directory entry. . . . .	unlink(2)
rm, rmdir:	remove files or directories. . . . .	rm(1)
eqn constructs. deroff:	remove nroff/troff, lbl, and . . . . .	deroff(1)
bits. strip5.0:	remove symbols and relocation . . . . .	strip5.0(1)
from a queue. insque,	remque: insert/remove element . . . . .	insque(3N)
remsh: remote shell. . . . .	remsh: remote shell. . . . .	remsh(1N)
remshd: remote shell server. . . . .	remshd: remote shell server. . . . .	remshd(8N)
check and interactive	repair. /system consistency . . . . .	fsock(1M)
uniq: report	repeated lines in a file. . . . .	uniq(1)
clock:	report CPU time used. . . . .	clock(3C)
communication/ ipcsc:	report inter-process . . . . .	ipcsc(1)
blocks. df:	report number of free disk . . . . .	df(1M)
errpt: process a	report of logged errors. . . . .	errpt(1M)
frequencies in a file. freq:	report on character . . . . .	freq(1)
sa2, sadc: system activity	report package. sa1, . . . . .	sa1(1M)
timex: time a command;	report process data and system/ . . . . .	timex(1)
ps:	report process status. . . . .	ps(1)

file. uniq:	report repeated lines in a	uniq(1)
sar: system activity	reporter.	sar(1)
files. version:	reports version number of	version(1)
stream. fseek, rewind, ftell:	reposition a file pointer in a	fseek(3S)
/lpmove: start/stop the LP	request scheduler and move/	lpsched(1M)
reject: allow/prevent LP	requests. accept,	accept(1M)
LP request scheduler and move	requests. /start/stop the	lpsched(1M)
lp, cancel: send/cancel	requests to an LP line/	lp(1)
teletype bits to a/ tset,	reset: set or reset the	tset(1)
sensible/ tset, reset: set or	reset the teletype bits to a	tset(1)
arp: A address	Resolution Protocol.	arp(5P)
object file. ldgetname:	retrieve symbol name for	ldgetname(3X)
argument. getarg:	return Fortran command-line	getarg(3F)
variable. getenv:	return Fortran environment	getenv(3F)
accounting. mclock:	return Fortran time	mclock(3F)
abs:	return integer absolute value.	abs(3C)
string. len:	return length of Fortran	len(3F)
substring. index:	return location of Fortran	index(3F)
logname:	return login name of user.	logname(3X)
command. rexec:	return stream to a remote	rexec(3N)
name. getenv:	return value for environment	getenv(3C)
stat: data	returned by stat system call.	stat(5)
/ruserok: routines for	returning a stream to a remote/	rcmd(3N)
configuration/ uvar:	returns system-specific	uvar(2)
col: filter	reverse line-feeds.	col(1)
file pointer in a/ fseek,	rewind, ftell: reposition a	fseek(3S)
/readdir, telldir, seekdir,	rewinddir, closedir: flexible/	directory(3X)
creat: create a new file or	rewrite an existing one.	creat(2)
remote command.	rexec: return stream to a	rexec(3N)
server.	rexecd: remote execution	rexecd(8N)
	rlogin: remote login.	rlogin(1N)
directories.	rlogind: remote login server.	rlogind(8N)
read mail. mail,	rm, rmdir: remove files or	rm(1)
SCCS file.	rmail: send mail to users or	mail(1)
directories. rm,	rmdel: remove a delta from an	rmdel(1)
Escape from the automatic	rmdir: remove files or	rm(1)
Try to escape the killer	robots. autorobots:	autorobots(6)
robots.	robots. chase:	chase(6)
robots: Escape from the	robots: Escape from the	robots(6)
robots.	robots.	robots(6)
chroot: change	root directory.	chroot(2)
chroot: change	root directory for a command.	chroot(1M)
logarithm, power, square	root functions. /exponential,	exp(3M)
/dsqrt, csqrt: Fortran square	root intrinsic function.	sqrt(3F)
routing tables.	route: manually manipulate the	route(8N)
daemon.	routed: network routing	routed(8N)
rcmd, rresvport, ruserok:	routines for returning a/	rcmd(3N)
Internet address manipulation	routines. /inet_netof:	inet(3N)
common object file access	routines. ldfcn:	ldfcn(4)
expression compile and match	routines. regexp: regular	regexp(5)
terminal independent operation	routines. /goto, tputs:	termcap(3X)
routed: network	routing daemon.	routed(8N)
route: manually manipulate the	routing tables.	route(8N)
for returning a stream/ rcmd,	rresvport, ruserok: routines	rcmd(3N)
standard/restricted/ sh,	rsh: shell, the	sh(1)
and, or, xor, not, lshift,	rshift: Fortran bitwise/	bool(3F)
nice:	run a command at low priority.	nice(1)
hangups (sh only). nohup:	run a command immune to	nohup(1)
runacct:	run daily accounting.	runacct(1M)
	runacct: run daily accounting.	runacct(1M)
/prctmp, prdaily, prtacct,	runacct, shutacct, startup,/	acctsh(1M)
local machines.	ruptime: show host status of	ruptime(1N)
returning a/ rcmd, rresvport,	ruserok: routines for	rcmd(3N)
machines.	rwho: who's logged in on local	rwho(1N)

	rwod: system status server. . . . .	rwod(8N)
activity report package.	sa1, sa2, sacd: system . . . . .	sar(1M)
report package. sa1,	sa2, sacd: system activity . . . . .	sar(1M)
editing activity.	sact: print current SCCS file . . . . .	sact(1)
package. sa1, sa2,	sacd: system activity report . . . . .	sar(1M)
	sag: system activity graph. . . . .	sag(1G)
	sar: system activity reporter. . . . .	sar(1)
space allocation. brk,	sbrk: change data segment . . . . .	brk(2)
formatted input.	scanf, fscanf, sscanf: convert . . . . .	scanf(3S)
bfs: big file	scanner. . . . .	bfs(1)
language. awk: pattern	scanning and processing . . . . .	awk(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 del: remove a delta from an	SCCS file. . . . .	rm del(1)
compare two versions of an	SCCS file. sccsdiff: . . . . .	sccsdiff(1)
sccsfile: format of	SCCS file. . . . .	sccsfile(4)
undo a previous get of an	SCCS file. unget: . . . . .	unget(1)
val: validate	SCCS file. . . . .	val(1)
admin: create and administer	SCCS files. . . . .	admin(1)
what: identify	SCCS files. . . . .	what(1)
help: ask for help in using	SCCS. . . . .	help(1)
of an SCCS file.	sccsdiff: compare two versions . . . . .	sccsdiff(1)
	sccsfile: format of SCCS file. . . . .	sccsfile(4)
/start/stop the LP request	scheduler and move requests. . . . .	lpsched(1M)
common object file.	scnhdr: section header for a . . . . .	scnhdr(4)
clear: clear terminal	screen. . . . .	clear(1)
optimization/ curses: CRT	screen handling and . . . . .	curses(3X)
twinkle: twinkle stars on the	screen. . . . .	twinkle(6)
display editor based on/ vi:	screen-oriented (visual) . . . . .	vi(1)
inittab:	script for the init process. . . . .	inittab(4)
terminal session.	script: make typescript of . . . . .	script(1)
system initialization shell	scripts. /rc, powerfail: . . . . .	brc(1M)
	sdb: symbolic debugger. . . . .	sdb(1)
program.	sdiff: side-by-side difference . . . . .	sdiff(1)
grep, egrep, fgrep	search a file for a pattern. . . . .	grep(1)
bsearch: binary	search a sorted table. . . . .	bsearch(3C)
accounting file(s). acctcom:	search and print process . . . . .	acctcom(1)
lsearch, lfind: linear	search and update. . . . .	lsearch(3C)
hcreate, hdestroy: manage hash	search tables. hsearch, . . . . .	hsearch(3C)
tdelete, twalk: manage binary	search trees. tsearch, tfind, . . . . .	tsearch(3C)
object file. scnhdr:	section header for a common . . . . .	scnhdr(4)
object/ /read an indexed/named	section header of a common . . . . .	ldhread(3X)
/to line number entries of a	section of a common object/ . . . . .	ldlseek(3X)
/to relocation entries of a	section of a common object/ . . . . .	ldrseek(3X)
/seek to an indexed/named	section of a common object/ . . . . .	ldsseek(3X)
files. size: print	section sizes of common object . . . . .	size(1)
	sed: stream editor. . . . .	sed(1)
/mrand48, jrand48, srand48,	seed48, lcong48: generate/ . . . . .	drand48(3C)
section of/ ldsseek, ldsseek:	seek to an indexed/named . . . . .	ldsseek(3X)
a section/ ldlseek, ldlseek:	seek to line number entries of . . . . .	ldlseek(3X)
a section/ ldrseek, ldrseek:	seek to relocation entries of . . . . .	ldrseek(3X)
header of a common/ ldohseek:	seek to the optional file . . . . .	ldohseek(3X)
common object file. ldtbseek:	seek to the symbol table of a . . . . .	ldtbseek(3X)
opendir, readdir, telldir,	seekdir, rewinddir, closedir:/ . . . . .	directory(3X)
shmget: get shared memory	segment. . . . .	shmget(2)
brk, sbrk: change data	segment space allocation. . . . .	brk(2)
to two sorted files. comm:	select or reject lines common . . . . .	comm(1)
multiplexing.	select: synchronous i/o . . . . .	select(2N)
greek:	select terminal filter. . . . .	greek(1)
of a file. cut: cut out	selected fields of each line . . . . .	cut(1)

file. dump:	dump	selected parts of an object . . . . .	dump(1)
semctl:	semctl	semaphore control operations. . . . .	semctl(2)
semop:	semop	semaphore operations. . . . .	semop(2)
ipcrm: remove a message queue,	semopset	semaphore set or shared memory/	ipcrm(1)
semget: get set of	semopset	semaphores. . . . .	semget(2)
operations.	semopset	semctl: semaphore control . . . . .	semctl(2)
send, sendto, sendmsg:	semopset	semget: get set of semaphores. . . . .	semget(2)
a group of processes. kill:	semopset	semop: semaphore operations. . . . .	semop(2)
mail. mail, rmail:	semopset	send a message from a socket. . . . .	send(2N)
message from a socket.	semopset	send a signal to a process or . . . . .	kill(2)
group. killpg:	semopset	send mail to users or read . . . . .	mail(1)
line printer. lp, cancel:	semopset	send, sendto, sendmsg: send a . . . . .	send(2N)
socket. send, sendto,	semopset	send signal to a process . . . . .	killpg(3N)
message from a socket. send,	semopset	send/cancel requests to an LP . . . . .	lp(1)
reset the teletype bits to a	semopset	sendmsg: send a message from a . . . . .	send(2N)
File Transfer Protocol	semopset	sendto, sendmsg: send a . . . . .	send(2N)
remshd: remote shell	semopset	sensible state. /reset: set or . . . . .	tset(1)
rexecd: remote execution	semopset	server. ftpd: DARPA Internet . . . . .	ftpd(8N)
rlogind: remote login	semopset	server. . . . .	remshd(8N)
rwhod: system status	semopset	server. . . . .	rexecd(8N)
telnetd: DARPA TELNET protocol	semopset	server. . . . .	rlogind(8N)
Trivial File Transfer Protocol	semopset	server. . . . .	rwhod(8N)
make typescript of terminal	semopset	server. tftpd: DARPA . . . . .	telnetd(8N)
buffering to a stream.	semopset	session. script: . . . . .	tftpd(8N)
IDs. setuid,	semopset	setbuf, setvbuf: assign . . . . .	script(1)
getgrent, getgrgid, getgrnam,	semopset	setgid: set user and group . . . . .	setbuf(3S)
/gethostbyaddr, gethostbyname,	semopset	setgrent, endgrent, fgetgrent:/ . . . . .	setuid(2)
identifier of/ gethostid,	semopset	sethostent, endhostent: get/ . . . . .	getgrent(3C)
current host. gethostname,	semopset	sethostid: get/set unique . . . . .	gethostent(2N)
goto.	semopset	sethostname: get/set name of . . . . .	gethostid(3N)
encryption. crypt,	semopset	setjmp, longjmp: non-local . . . . .	gethostname(2N)
/getnetbyaddr, getnetbyname,	semopset	setkey, encrypt: generate DES . . . . .	setjmp(3C)
protocol/ /getprotobyname,	semopset	setmnt: establish mount table. . . . .	crypt(3C)
getpwent, getpwuid, getpwnam,	semopset	setnetent, endnetent: get/ . . . . .	setmnt(1M)
effective group ID.	semopset	setpgrp: set process group ID. . . . .	getnetent(3N)
effective user ID's.	semopset	setprotoent, endprotoent: get . . . . .	setpgrp(2)
/getservbyport, getservbyname,	semopset	setpwent, endpwent, fgetpwent:/ . . . . .	setprotoent(3N)
options on/ getsockopt,	semopset	setregid: set real and . . . . .	getpwent(3C)
login time. profile:	semopset	setreuid: set real and . . . . .	setregid(2)
gettydefs: speed and terminal	semopset	setservent, endservent: get/ . . . . .	setreuid(2)
disktune: tune floppy disk	semopset	setsockopt: get and set . . . . .	setservent(3N)
group IDs.	semopset	setting up an environment at . . . . .	setsockopt(2N)
/getutid, getutline, pututline,	semopset	settings used by getty. . . . .	profile(4)
stream. setbuf,	semopset	settling time parameters. . . . .	gettydefs(4)
data in a machine/ spull,	semopset	setuid, setgid: set user and . . . . .	disktune(1M)
a command immune to hangups	semopset	setutent, endutent, utmpname:/ . . . . .	setuid(2)
standard/restricted command/	semopset	setvbuf: assign buffering to a . . . . .	setut(3C)
operations. shmctl:	semopset	setgid: access long integer . . . . .	setvbuf(3S)
queue, semaphore set or	semopset	(sh only). nohup: run . . . . .	spull(3X)
shmop:	semopset	sh, rsh: shell, the . . . . .	nohup(1)
shmget: get	semopset	shared memory control . . . . .	sh(1)
from C programs to implement	semopset	shared memory id. /a message . . . . .	shmctl(2)
system: issue a	semopset	shared memory operations. . . . .	ipcrm(1)
with C-like syntax. csh: a	semopset	shared memory segment. . . . .	shmop(2)
system: issue a	semopset	shared strings. /strings . . . . .	shmget(2)
sh:	semopset	shell command from Fortran. . . . .	xstr(1)
shutacct, startup, turnacct:	semopset	shell (command interpreter) . . . . .	system(3F)
remsh: remote	semopset	shell command. . . . .	csh(1)
system initialization	semopset	shell layer manager. . . . .	system(3S)
remshd: remote	semopset	shell procedures for/ /runacct, . . . . .	sh(1)
	semopset	shell. . . . .	acctsh(1M)
	semopset	shell scripts. /rc, powerfail: . . . . .	remsh(1N)
	semopset	shell server. . . . .	brc(1M)
	semopset		remshd(8N)

Permuted Index

command programming/ sh, rsh: shell, the standard/restricted . . . . . sh(1)  
 sh: shell layer manager. . . . . shl(1)  
     operations. shmctl: shared memory control . . . . . shmctl(2)  
     segment. shmget: get shared memory . . . . . shmget(2)  
     operations. shmop: shared memory . . . . . shmop(2)  
 full-duplex/ shutdown: shut down part of a . . . . . shutdown(2N)  
 /prdaily, prtacct, runacct, shutacct, startup, turnacct:/ . . . . . acctsh(1M)  
 full-duplex connection. shutdown: shut down part of a . . . . . shutdown(2N)  
     processing. shutdown: terminate all . . . . . shutdown(1M)  
     program. sdiff: side-by-side difference . . . . . sdiff(1)  
 transfer-of-sign intrinsic/ sign, isign, dsign: Fortran . . . . . sign(3F)  
     login: sign on. . . . . login(1)  
 pause: suspend process until signal. . . . . pause(2)  
 what to do upon receipt of a signal. signal: specify . . . . . signal(2)  
 action on receipt of a system signal. /specify Fortran . . . . . signal(3F)  
     on receipt of a system/ signal: specify Fortran action . . . . . signal(3F)  
     upon receipt of a signal. signal: specify what to do . . . . . signal(2)  
     killpg: send signal to a process group. . . . . killpg(3N)  
     of processes. kill: send signal to a process or a group . . . . . kill(2)  
     asignal, gsignal: software signals. . . . . signal(3C)  
 lex: generate programs for simple lexical tasks. . . . . lex(1)  
 generator. rand, srand: simple random-number . . . . . rand(3C)  
 tc: phototypesetter simulator. . . . . tc(1)  
 atan, atan2: trigonometric/ sin, cos, tan, asin, acos, . . . . . trig(3M)  
 intrinsic function. sin, dsin, csin: Fortran sine . . . . . sin(3F)  
 sin, dsin, csin: Fortran sine intrinsic function. . . . . sin(3F)  
 /dsinh: Fortran hyperbolic sine intrinsic function. . . . . sinh(3F)  
     ssp: make output single spaced. . . . . asp(1)  
     functions. sinh, cosh, tanh: hyperbolic . . . . . sinh(3M)  
 hyperbolic sine intrinsic/ sinh, dsinh: Fortran . . . . . sinh(3F)  
     get descriptor table size. getdtablesize: . . . . . getdtablesize(3N)  
     size5.0: size of an object file. . . . . size5.0(1)  
 common object files. size: print section sizes of . . . . . size(1)  
     file. size5.0: size of an object . . . . . size5.0(1)  
     size: print section sizes of common object files. . . . . size(1)  
     interval. sleep: suspend execution for . . . . . sleep(1)  
     interval. sleep: suspend execution for . . . . . sleep(3C)  
 documents, view graphs, and slides. mmt, mvt: typeset . . . . . mmt(1)  
 typesetting view graphs and slides. /macro package for . . . . . mv(5)  
 current/ ttypslot: find the slot in the utmp file of the . . . . . ttypslot(3C)  
 spline: interpolate smooth curve. . . . . spline(1G)  
 int, ifix, idint, real, float, snpl, dble, cmplx, dcmplx,/ . . . . . ftype(3F)  
     sno: SNOBOL interpreter. . . . . sno(1)  
     SNOBOL interpreter. . . . . sno(1)  
     accept a connection on a socket. accept: . . . . . accept(2N)  
     bind: bind a name to a socket. . . . . bind(2N)  
     initiate a connection on a socket. connect: . . . . . connect(2N)  
     communication. socket: create an endpoint for . . . . . socket(2N)  
 listen for connections on a socket. listen: . . . . . listen(2N)  
     getsockname: get socket name. . . . . getsockname(2N)  
 receive a message from a socket. /recvfrom, recvmsg: . . . . . recv(2N)  
 sendmsg: send a message from a socket. send, sendto, . . . . . send(2N)  
     get and set options on sockets. /setsockopt: . . . . . getsockopt(2N)  
 pg: file perusal filter for soft-copy terminals. . . . . pg(1)  
 interface. lo: software loopback network . . . . . lo(5)  
     ssignal, gsignal: software signals. . . . . signal(3C)  
     sort: sort and/or merge files. . . . . sort(1)  
     qsort: quicker sort. . . . . qsort(3C)  
     sort: sort and/or merge files. . . . . sort(1)  
     tsort: topological sort. . . . . tsort(1)  
 or reject lines common to two sorted files. comm: select . . . . . comm(1)  
     bsearch: binary search a sorted table. . . . . bsearch(3C)  
     for program. whereis: locate source, binary, and/or manual . . . . . whereis(1)  
     message file by massaging C source. /create an error . . . . . mkstr(1)

brk, sbrk: change data segment	space allocation.	brk(2)
ssp: make output single	spaced.	ssp(1)
terminal. ct:	spawn getty to a remote	ct(1C)
fspec: format	specification in text files.	fspec(4)
the extended errors in the	specified device. /turn on/off	exterr(1)
receipt of a system/ signal:	specify Fortran action on	signal(3F)
receipt of a signal. signal:	specify what to do upon	signal(2)
/set terminal type, modes,	speed, and line discipline.	getty(1M)
used by getty. gettydefs:	speed and terminal settings	gettydefs(4)
hashcheck: find spelling/	spell, hashmake, spellin,	spell(1)
spelling/ spell, hashmake,	spellin, hashcheck: find	spell(1)
spellin, hashcheck: find	spelling errors. /hashmake,	spell(1)
curve.	spline: interpolate smooth	spline(1G)
split:	split a file into pieces.	split(1)
csplit: context	split.	csplit(1)
files. fsplit:	split f77, ratfor, or elf	fsplit(1)
pieces.	split: split a file into	split(1)
uuclean: uucp	spool directory clean-up.	uuclean(1M)
lpadmin: configure the LP	spooling system.	lpadmin(1M)
output. printf, fprintf,	sprintf: print formatted	printf(3S)
integer data in a machine/	sputl, sgetl: access long	sputl(3X)
square root intrinsic/	sqr, dsqr, csqr: Fortran	sqr(3F)
power./ exp, log, log10, pow,	sqr: exponential, logarithm,	exp(3M)
exponential, logarithm, power,	square root functions. /sqr:	exp(3M)
sqr, dsqr, csqr: Fortran	square root intrinsic/	sqr(3F)
random-number/ irand,	strand, rand: Fortran uniform	rand(3F)
generator. rand,	strand: simple random-number	rand(3C)
/nrand48, mrand48, irand48,	strand48, seed48, lcong48:/	drand48(3C)
rcvhex: translates Motorola	S-records from downloading/	rcvhex(1)
input. scanf, fscanf,	sscanf: convert formatted	scanf(3S)
signals.	ssignal, gsignal: software	ssignal(3C)
spaced.	ssp: make output single	ssp(1)
package. stdio:	standard buffered input/output	stdio(3S)
communication package. ftok:	standard interprocess	stdipc(3C)
sh, rsh: shell, the	standard/restricted command/	sh(1)
twinkle: twinkle	stars on the screen.	twinkle(6)
lpsched, lpshut, lpmove:	start/stop the LP request/	lpsched(1M)
boot:	startup procedures.	boot(8)
/prtacct, runacct, shutacct,	startup, turnacct: shell/	acctsh(1M)
system call.	stat: data returned by stat	stat(5)
	stat, fstat: get file status.	stat(2)
stat: data returned by	stat system call.	stat(5)
ff: list file names and	statistics for a file system.	ff(1M)
lav: print load average	statistics.	lav(1)
ustat: get file system	statistics.	ustat(2)
lpstat: print LP	status information.	lpstat(1)
feof, clearerr, fileno: stream	status inquiries. ferror,	ferror(3S)
control. uustat: uucp	status inquiry and job	uustat(1C)
communication facilities	status. /report inter-process	ipcs(1)
netstat: show network	status.	netstat(1N)
ruptime: show host	status of local machines.	ruptime(1N)
ps: report process	status.	ps(1)
rwthod: system	status server.	rwthod(8N)
stat, fstat: get file	status.	stat(2)
input/output package.	stdio: standard buffered	stdio(3S)
	stime: set time.	stime(2)
wait for child process to	stop or terminate. wait:	wait(2)
wait for child process to	stop or terminate. wait3:	wait3(2N)
strncmp, strcpy, strncpy,/	strcat, strncat, strcmp,	string(3C)
/strcpy, strncpy, strlen,	strchr, strrchr, strpbrk,/	string(3C)
strncpy,/ strcat, strncat,	strcmp, strncmp, strcpy,	string(3C)
/strncat, strcmp, strncmp,	strcpy, strncpy, strlen,/	string(3C)
/strchr, strpbrk, strstrn:	strcspn, strtok: string/	string(3C)
sed:	stream editor.	sed(1)

fflush: close or flush a  
 fopen, freopen, fdopen: open a  
     reposition a file pointer in a  
     get character or word from a  
     fgets: get a string from a  
     put character or word on a  
     puts, fputs: put a string on a  
     setvbuf: assign buffering to a  
     /feof, clearerr, fileno:  
     /routines for returning a  
     rexec: return  
 push character back into input  
 long integer and base-64 ASCII  
     lge, lgt, lle, llt:  
     convert date and time to  
     floating-point number to  
     gets, fgets: get a  
 len: return length of Fortran  
     puts, fputs: put a  
 bcmp, bzero, ffs: bit and byte  
     strspn, strcspn, strtok:  
 number. strtod, atof: convert  
 number. atof: convert ASCII  
 strtok, atol, atoi: convert  
     strings in an object, or/  
     implement/ xstr: extract  
     strings: find the printable  
 C programs to implement shared  
 number information from an/  
 information from an/ strip:  
     relocation bits.  
     /strncmp, strcpy, strncpy,  
     strcpy, strncpy,/ strcat,  
     strcat, strcat, strcmp,  
     /strcmp, strncmp, strcpy,  
     /strlen, strchr, strrchr,  
     /strncpy, strlen, strchr,  
     /strchr, strchr, strpbrk,  
     to double-precision number.  
     /strpbrk, strspn, strcspn,  
     string to integer.  
 processes using a file or file  
     terminal.  
     another user.  
     intro: introduction to  
     plot: graphics interface  
 /same lines of several files or  
 return location of Fortran  
     file. sum7:  
     the files in the/ sumdir:  
     count of a file.  
     a file.  
     characters in the files in/  
     du:  
 accounting/ acctcms: command  
     sync: update the  
     sync: update  
     su: become  
     interval. sleep:  
     interval. sleep:  
     pause:  
     swab:  
     swab:  
     sxt: pseudo-device driver.

fclose (3S)  
 fopen (3S)  
 fseek (3S)  
 getc (3S)  
 gets (3S)  
 putc (3S)  
 puts (3S)  
 setbuf (3S)  
 fflush (3S)  
 rcmd (3N)  
 rexec (3N)  
 ungetc (3S)  
 a64l (3C)  
 strcmp (3F)  
 ctime (3C)  
 ecvt (3C)  
 gets (3S)  
 len (3F)  
 puts (3S)  
 bstring (3N)  
 string (3C)  
 strtod (3C)  
 atof (3C)  
 strtol (3C)  
 strings (1)  
 xstr (1)  
 strings (1)  
 xstr (1)  
 strip (1)  
 strip (1)  
 strip5.0 (1)  
 string (3C)  
 string (3C)  
 string (3C)  
 string (3C)  
 string (3C)  
 string (3C)  
 string (3C)  
 strtod (3C)  
 string (3C)  
 strtol (3C)  
 fuser (1M)  
 stty (1)  
 su (1)  
 intro (3)  
 plot (3X)  
 paste (1)  
 index (3F)  
 sum7 (1)  
 sumdir (1)  
 sum (1)  
 sum7 (1)  
 sumdir (1)  
 du (1)  
 acctcms (1M)  
 sync (1)  
 sync (2)  
 su (1)  
 sleep (1)  
 sleep (3C)  
 pause (2)  
 swab (3C)  
 swab (3C)  
 sxt (7)



information from/ strip:	strip	strip(1)
ldgetname:	retrieve	ldgetname(3X)
object/ /compute the index of a	symbol name for object file	ldtbindx(3X)
ldtread:	symbol table entry of a common	ldtread(3X)
symbol:	symbol table entry of a common/	syms(4)
object/ ldtbseek:	symbol table format	ldtbseek(3X)
	symbol table of a common	sdb(1)
	symbolic debugger	strip5.0(1)
strip5.0:	symbols and relocation bits	syms(4)
symbol table format	syms: common object file	sync(2)
	sync: update super-block	sync(1)
	sync: update the super block	select(2N)
select:	synchronous i/o multiplexing	cs(1)
interpreter) with C-like	syntax. csh: a shell (command	sysdef(1M)
	sysdef: system definition	perror(3C)
error/ perror, errno,	sys_errlist, sys_nerr: system	perror(3C)
perror, errno, sys_errlist,	sys_nerr: system error/	uvar(2)
information. uvar: returns	system-specific configuration	bsearch(3C)
binary search a sorted	table. bsearch:	ldtbindx(3X)
/compute the index of a symbol	table entry of a common object/	ldtread(3X)
file. /read an indexed symbol	table entry of a common object	syms(4)
common object file symbol	table format. syms:	master(4)
master device information	table. master:	mnttab(4)
mnttab: mounted file system	table.	ldtbseek(3X)
ldtbseek: seek to the symbol	table of a common object file.	setmnt(1M)
setmnt: establish mount	table.	getdtablesize(3N)
getdtablesize: get descriptor	table size.	t(1)
tbl: format	tables for nroff or troff.	hsearch(3C)
hdestroy: manage hash search	tables. hsearch, hcreate,	route(8N)
manipulate the routing	tables. route: manually	tabs(1)
tabs: set	tabs on a terminal.	tabs(1)
	tabs: set tabs on a terminal.	ctags(1)
ctags: maintain a	tags file for a C program.	tail(1)
a file.	tail: deliver the last part of	take(1C)
remote machine.	take: takes a file from a	take(1C)
machine. take:	takes a file from a remote	talk(1N)
	talk: talk to another user.	talk(1N)
talk:	talk to another user.	trig(3M)
trigonometric/ sin, cos,	tan, asin, acos, atan, atan2:	tan(3F)
intrinsic function.	tan, dtan: Fortran tangent	tan(3F)
tan, dtan: Fortran	langent intrinsic function.	tanh(3F)
/dtanh: Fortran hyperbolic	tangent intrinsic function.	tanh(3F)
hyperbolic tangent intrinsic/	tanh, dtanh: Fortran	sinh(3M)
sinh, cosh,	tanh: hyperbolic functions.	tp(1)
tp: manipulate	tape archive.	tar(1)
tar:	tape file archiver.	frec(1M)
recover files from a backup	tape. frec:	filesave(1M)
file system backup. filesave,	tapasave: daily/weekly UNIX	tar(1)
	tar: tape file archiver.	lex(1)
programs for simple lexical	tasks. lex: generate	deroff(1)
deroff: remove nroff/troff,	tbl, and eqn constructs.	tbl(1)
or troff.	tbl: format tables for nroff	tc(1)
	tc: phototypesetter simulator.	tcp(5P)
Control Protocol.	tcp: Internet Transmission	tdelete, twalk: manage binary
search trees. tsearch, tfind,	tdelete, twalk: manage binary	tee: pipe fitting.
	tee: pipe fitting.	Tektronix 4014 terminal.
4014: paginator for the	Tektronix 4014 terminal.	teletype bits to a sensible/
tset, reset: set or reset the	teletype bits to a sensible/	teletypes. last: indicate
last logins of users and	teletypes. last: indicate	telinit: process control
initialization. init,	telinit: process control	telldir, seekdir, rewinddir,
closedir:/ opendir, readdir,	telldir, seekdir, rewinddir,	TELNET protocol server.
telnetd: DARPA	TELNET protocol server.	TELNET protocol.
telnet: user interface to the	TELNET protocol.	telnet: user interface to the
TELNET protocol.	telnet: user interface to the	telnetd: DARPA TELNET protocol
server.	telnetd: DARPA TELNET protocol	

temporary file. tmpnam,	tempnam: create a name for a	tmpnam(3S)
tmpfile: create a	temporary file.	tmpfile(3S)
tempnam: create a name for a	temporary file. tmpnam,	tmpnam(3S)
terminals.	term: conventional names for	term(5)
term: format of compiled	term file..	term(4)
file..	term: format of compiled term	term(4)
data base.	termcap: terminal capability	termcap(5)
for the Tektronix 4014	terminal. 4014: paginator	4014(1)
functions of the DASI 450	terminal. 450: handle special	450(1)
termcap:	terminal capability data base.	termcap(5)
terminfo:	terminal capability data base.	terminfo(4)
ct: spawn getty to a remote	terminal.	ct(1C)
ctermid: generate filename for	terminal.	ctermid(3S)
pty: pseudo	terminal driver.	pty(5)
greek: select	terminal filter.	greek(1)
/tgetstr, tgoto, tputs:	terminal independent operation/	termcap(3X)
termio: general	terminal interface.	termio(7)
tty: controlling	terminal interface.	tty(7)
dial: establish an out-going	terminal line connection.	dial(3C)
clear: clear	terminal screen.	clear(1)
script: make typescript of	terminal session.	script(1)
getty. gettydefs: speed and	terminal settings used by	gettydefs(4)
stty: set the options for a	terminal.	stty(1)
tabs: set tabs on a	terminal.	tabs(1)
isatty: find name of a	terminal. ttyname,	ttyname(3C)
and line/ getty: set	terminal type, modes, speed,	getty(1M)
ttytype: data base of	terminal types by port.	ttytype(4)
animate worms on a display	terminal. worms:	worms(6)
functions of DASI 300 and 300s	terminals. /handle special	300(1)
tty: get the	terminal's name.	tty(1)
perusal filter for soft-copy	terminals. pg: file	pg(1)
term: conventional names for	terminals.	term(5)
kill:	terminate a process.	kill(1)
shutdown:	terminate all processing.	shutdown(1M)
abort:	terminate Fortran program.	abort(3F)
exit, _exit:	terminate process.	exit(2)
daemon. errstop:	terminate the error-logging	errstop(1M)
for child process to stop or	terminate. wait: wait	wait(2)
for child process to stop or	terminate. wait3: wait	wait3(2N)
tic:	terminfo compiler.	tic(1M)
tput: query	terminfo database.	tput(1)
data base.	terminfo: terminal capability	terminfo(4)
interface.	termio: general terminal	termio(7)
command.	test: condition evaluation	test(1)
quiz:	test your knowledge.	quiz(6)
ed, red:	text editor.	ed(1)
ex, edit:	text editor.	ex(1)
change the format of a	text file. newform:	newform(1)
fspec: format specification in	text files.	fspec(4)
/checkeq: format mathematical	text for nroff or troff.	eqn(1)
prepare constant-width	text for troff. cw, checkcw:	cw(1)
nroff: format	text.	nroff(1)
plock: lock process,	text, or data in memory.	plock(2)
troff: typeset	text.	troff(1)
binary search trees. tsearch,	tfind, tdelete, twalk: manage	tsearch(3C)
Transfer Protocol server.	tftp: DARPA Trivial File	tftp(8N)
tgetstr, tgoto, tputs:/	tgetent, tgetnum, tgetflag,	termcap(3X)
tputs:/ tgetent, tgetnum,	tgetflag, tgetstr, tgoto,	termcap(3X)
tgoto, tputs:/ tgetent,	tgetnum, tgetflag, tgetstr,	termcap(3X)
tgetent, tgetnum, tgetflag,	tgetstr, tgoto, tputs:/	termcap(3X)
/tgetnum, tgetflag, tgetstr,	tgoto, tputs: terminal/	termcap(3X)
	tic: terminfo compiler.	tic(1M)
ttt, cubic:	tic-tac-toe.	ttt(6)
data and system/ timex:	time a command; report process	timex(1)

time:	time a command.	time(1)
mclock: return Fortran	time accounting.	mclock(3F)
execute commands at a later	time. at, batch:	at(1)
systems for optimal access	time. dcopy: copy file	dcopy(1M)
	time: get time.	time(2)
tune floppy disk settling	time parameters. disktime:	disktime(1M)
profit: execution	time profile.	profil(2)
up an environment at login	time. profile: setting	profile(4)
stime: set	time.	stime(2)
	time: time a command.	time(1)
time: get	time.	time(2)
tzset: convert date and	time to string. /asctime,	ctime(3C)
clock: report CPU	time used.	clock(3C)
process times.	times: get process and child	times(2)
update access and modification	times of a file. touch:	touch(1)
get process and child process	times. times:	times(2)
file access and modification	times. utime: set	utime(2)
process data and system/	timex: time a command; report	timex(1)
file.	tmpfile: create a temporary	tmpfile(3S)
for a temporary file.	tmpnam, tmpnam: create a name	tmpnam(3S)
/tolower, _toupper, _tolower,	toascii: translate characters.	conv(3C)
popen, pclose: initiate pipe	to/from a process.	popen(3S)
toupper, tolower, _toupper,	_tolower, toascii: translate/	conv(3C)
toascii: translate/ toupper,	tolower, _toupper, _tolower,	conv(3C)
tsort:	topological sort.	tsort(1)
acctmrg: merge or add	total accounting files.	acctmrg(1M)
modification times of a file.	touch: update access and	touch(1)
translate/ toupper, tolower,	_toupper, _tolower, toascii:	conv(3C)
_tolower, toascii: translate/	toupper, tolower, _toupper,	conv(3C)
	tp: manipulate tape archive.	tp(1)
	tplot: graphics filters.	tplot(1G)
	tput: query terminfo database.	tput(1)
/tgetflag, tgetstr, tgoto,	tputs: terminal independent/	termcap(3X)
	tr: translate characters.	tr(1)
ptrace: process	trace.	ptrace(2)
trpt: transliterate protocol	trace.	trpt(8N)
bit, bit512: block	transfer data.	bit(3C)
ftp: file	transfer program.	ftp(1N)
ftpd: DARPA Internet File	Transfer Protocol server.	ftpd(8N)
tftpd: DARPA Trivial File	Transfer Protocol server.	tftpd(8N)
sign, isign, dsign: Fortran	transfer-of-sign intrinsic/	sign(3F)
/_toupper, _tolower, toascii:	translate characters.	conv(3C)
tr:	translate characters.	tr(1)
from downloading into/ rcvhex:	translates Motorola S-records	rcvhex(1)
hex:	translates object files.	hex(1)
trpt:	transliterate protocol trace.	trpt(8N)
tcp: Internet	Transmission Control Protocol.	tcp(5P)
ftw: walk a file	tree.	ftw(3C)
twalk: manage binary search	trees. /ifind, tdelete,	tsearch(3C)
	trek: trekkie game.	trek(6)
trek:	trekkie game.	trek(6)
tan, asin, acos, atan, atan2:	trigonometric functions. /cos,	trig(3M)
server. tftpd: DARPA	Trivial File Transfer Protocol	tftpd(8N)
constant-width text for	troff. cw, checkcw: prepare	cw(1)
mathematical text for nroff or	troff. /neqn, checkeq: format	eqn(1)
typesetting view graphs/ mv: a	troff macro package for	mv(5)
format tables for nroff or	troff. tbl:	tbl(1)
	troff: typeset text.	troff(1)
trace.	trpt: transliterate protocol	trpt(8N)
values.	true, false: provide truth	true(1)
pdp11, u3b, u3b5, vax: provide	truth value about your/ m68k,	machid(1)
true, false: provide	truth values.	true(1)
robots. chase:	Try to escape the killer	chase(6)
twalk: manage binary search/	tsearch, tfind, tdelete,	tsearch(3C)

*Permuted Index*

teletype bits to a sensible/ interface.	tset, reset: set or reset the . . . . .	tset(1)
	tsort: topological sort. . . . .	tsort(1)
	ttt, cubic: tic-tac-toe. . . . .	ttt(6)
	tty: controlling terminal . . . . .	tty(7)
	tty: get the terminal's name. . . . .	tty(1)
graphics for the extended a terminal.	TTY-37 type-box. greek: . . . . .	greek(5)
utmp file of the current/ types by port.	ttyname, isatty: find name of . . . . .	ttyname(3C)
parameters. disktime:	ttyslot: find the slot in the . . . . .	ttyslot(3C)
/runacct, shutacct, startup,	ttysize: data base of terminal . . . . .	ttysize(4)
tsearch, tfind, tdelete,	tune floppy disk settling time . . . . .	disktime(1M)
twinkle:	turnacct: shell procedures for/ . . . . .	acctsh(1M)
screen.	twalk: manage binary search/ . . . . .	tsearch(3C)
ichar, char: explicit Fortran file: determine file	twinkle stars on the screen. . . . .	twinkle(6)
value about your processor	twinkle: twinkle stars on the . . . . .	twinkle(6)
getty: set terminal	type conversion. /dcmplx, . . . . .	ftype(3F)
for the extended TTY-37	type. . . . .	file(1)
ttysize: data base of terminal	type. /vax: provide truth . . . . .	machid(1)
types.	type, modes, speed, and line/ . . . . .	getty(1M)
types: primitive system data	type-box. greek: graphics . . . . .	greek(5)
session. script: make	types by port. . . . .	ttysize(4)
graphs, and slides. mmt, mv:	types: primitive system data . . . . .	types(5)
troff:	types. . . . .	types(5)
mv: a troff macro package for	typescript of terminal . . . . .	script(1)
/localtime, gmtime, asctime,	typeset documents, view . . . . .	mmt(1)
value about your/ m68k, pdp11,	typeset text. . . . .	troff(1)
about your/ m68k, pdp11, u3b,	typesetting view graphs and/ . . . . .	mv(5)
Protocol.	tzset: convert date and time/ . . . . .	ctime(3C)
getpw: get name from	u3b, u3b5, vax: provide truth . . . . .	machid(1)
	u3b5, vax: provide truth value . . . . .	machid(1)
	udp: Internet User Datagram . . . . .	udp(5P)
	UID. . . . .	getpw(3C)
	ul: do underlining. . . . .	ul(1)
	ulimit: get and set user . . . . .	ulimit(2)
	umask: set and get file . . . . .	umask(2)
	umask: set file-creation mode . . . . .	umask(1)
file system. mount,	umount: mount and dismount . . . . .	mount(1M)
	umount: unmount a file system. . . . .	umount(2)
UNIX system.	uname: get name of current . . . . .	uname(2)
UNIX system.	uname: print name of current . . . . .	uname(1)
	underlining. . . . .	ul(1)
	undo a previous get of an SCCS . . . . .	unget(1)
	unget: undo a previous get of . . . . .	unget(1)
	ungetc: push character back . . . . .	ungetc(3S)
	uniform random-number/ . . . . .	rand(3F)
	uniformly distributed/ . . . . .	drand48(3C)
	uniq: report repeated lines in . . . . .	uniq(1)
	unique filename. . . . .	mktemp(3C)
	unique identifier of current/ . . . . .	gethostid(2N)
	units: conversion program. . . . .	units(1)
	UNIX-to-UNIX system command . . . . .	uux(1C)
	UNIX-to-UNIX system file copy. . . . .	uuto(1C)
	unlink: exercise link and . . . . .	link(1M)
	unlink: remove directory . . . . .	unlink(2)
	unlink system calls. link, . . . . .	link(1M)
	unmount a file system. . . . .	umount(2)
	unpack: compress and expand . . . . .	pack(1)
	update access and modification . . . . .	touch(1)
	update, and regenerate groups . . . . .	make(1)
	update bad block information. . . . .	badblk(1M)
	update files between two . . . . .	updater(1)
	update files between two . . . . .	updater(1M)
	update. tsearch, . . . . .	tsearch(3C)
	update super-block. . . . .	sync(2)
	update the super block. . . . .	sync(1)

two machines.	updater: update files between . . . .	updater(1)
two machines.	updater: update files between	updater(1M)
du: summarize disk	usage. . . . .	du(1)
id: print	user and group IDs and names.	id(1)
setuid, setgid: set	user and group IDs.	setuid(2)
crontab:	user crontab file. . . . .	crontab(1)
character login name of the	user. cuserid: get . . . . .	cuserid(3S)
udp: Internet	User Datagram Protocol. . . . .	udp(5P)
/getgid, getegid: get real	user, effective user, real/	getuid(2)
environ:	user environment. . . . .	environ(5)
disk accounting data by	user ID. diskusg: generate . . . . .	diskusg(1M)
print effective current	user id. whoami: . . . . .	whoami(1)
set real and effective	user ID's. setreuid: . . . . .	setreuid(2)
protocol. telnet:	user interface to the TELNET . . . . .	telnet(1N)
ulimit: get and set	user limits. . . . .	ulimit(2)
logname: return login name of	user. . . . .	logname(3X)
/get real user, effective	user, real group, and/	getuid(2)
become super-user or another	user. su: . . . . .	su(1)
talk: talk to another	user. . . . .	talk(1N)
the utmp file of the current	user. /find the slot in . . . . .	ttyslot(3C)
write: write to another	user. . . . .	write(1)
last: indicate last logins of	users and teletypes. . . . .	last(1)
mail, rmail: send mail to	users or read mail. . . . .	mail(1)
wall: write to all	users. . . . .	wall(1M)
fuser: identify processes	using a file or file/ . . . . .	fuser(1M)
help: ask for help in	using SCCS. . . . .	help(1)
statistics.	ustat: get file system . . . . .	ustat(2)
modification times.	utime: set file access and	utime(2)
utmp, wtmp:	utmp and wtmp entry formats.	utmp(4)
endutent, utmpname: access	utmp file entry. /setutent,	getut(3C)
ttyslot: find the slot in the	utmp file of the current user. . . . .	ttyslot(3C)
entry formats.	utmp, wtmp: utmp and wtmp	utmp(4)
/pututline, setutent, endutent,	utmpname: access utmp file/	getut(3C)
clean-up.	uuclean: uucp spool directory	uuclean(1M)
uusub: monitor	uucp network. . . . .	uusub(1M)
uuclean:	uucp spool directory clean-up.	uuclean(1M)
control. uustat:	uucp status inquiry and job	uustat(1C)
system to UNIX system copy.	uucp, uulog, uuname: UNIX	uucp(1C)
UNIX system copy. uucp,	uulog, uuname: UNIX system to	uucp(1C)
system copy. uucp, uulog,	uuname: UNIX system to UNIX	uucp(1C)
system file copy. uuto,	uupick: public UNIX-to-UNIX	uuto(1C)
and job control.	uustat: uucp status inquiry	uustat(1C)
UNIX-to-UNIX system file/	uusub: monitor uucp network.	uusub(1M)
command execution.	uuto, uupick: public . . . . .	uuto(1C)
configuration information.	uux: UNIX-to-UNIX system	uux(1C)
	uvar: returns system-specific	uvar(2)
	val: validate SCCS file. . . . .	val(1)
	validate SCCS file. . . . .	val(1)
	value about your processor/	machid(1)
/u3b, u3b5, vax: provide truth	value. . . . .	abs(3C)
abs: return integer absolute	value. abs, iabs, dabs, . . . . .	abs(3F)
cabs, zabs: Fortran absolute	value for environment name. . . . .	getenv(3C)
getenv: return	value functions. /fabs: floor,	floor(3M)
ceiling, remainder, absolute	value to environment. . . . .	putenv(3C)
putenv: change or add	values between host and/	byteorder(3N)
/htons, ntohl, ntohs: convert	values: machine-dependent	values(5)
values.	values. . . . .	true(1)
true, false: provide truth	values. . . . .	values(5)
values: machine-dependent	varargs argument list. . . . .	vprintf(3S)
/print formatted output of a	varargs argument list. . . . .	vprintf(3X)
/print formatted output of a	varargs: handle variable	varargs(5)
argument list.	variable argument list. . . . .	varargs(5)
varargs: handle	variable. getenv: . . . . .	getenv(3F)
return Fortran environment	vax: provide truth value about	machid(1)
your/ m68k, pdp11, u3b, u3b5,		

/files between M68000 and	VAX-11/780 processors.	fscv(1M)
	vc: version control.	vc(1)
	vchk: version checkup.	vchk(1M)
option letter from argument	vector. getopt: get	getopt(3C)
assert:	verify program assertion.	assert(3X)
vchk:	version checkup.	vchk(1M)
vc:	version control.	vc(1)
version: reports	version number of files.	version(1)
get: get a	version of an SCCS file.	get(1)
number of files.	version: reports version	version(1)
sccsdiff: compare two	versions of an SCCS file.	sccsdiff(1)
formatted output of/ vprintf,	vfprintf, vsprintf: print	vprintf(3S)
formatted output of/ vprintf,	vfprintf, vsprintf: print	vprintf(3X)
display editor based on ex.	vi: screen-oriented (visual)	vi(1)
mmt, mvt: typeset documents,	view graphs, and slides.	mmt(1)
macro package for typesetting	view graphs and slides. /troff	mv(5)
file perusal filter for crt	viewing. more:	more(1)
on ex. vi: screen-oriented	(visual) display editor based	vi(1)
systems with label checking.	volcopy, labelit: copy file	volcopy(1M)
file system: format of system	volume.	fs(4)
print formatted output of a/	vprintf, fprintf, vsprintf:	vprintf(3S)
print formatted output of a/	vprintf, fprintf, vsprintf:	vprintf(3X)
output of/ vprintf, fprintf,	vsprintf: print formatted	vprintf(3S)
output of/ vprintf, fprintf,	vsprintf: print formatted	vprintf(3X)
or terminate. wait:	wait for child process to stop	wait(2)
to stop or terminate. wait3:	wait for child process to stop	wait3(2N)
to stop or terminate.	wait: wait for child process	wait(2)
to stop or terminate.	wait3: wait for child process	wait3(2N)
ftw:	walk a file tree.	ftw(3C)
	wall: write to all users.	wall(1M)
	wc: word count.	wc(1)
	what: identify SCCS files.	what(1)
signal. signal: specify	what to do upon receipt of a	signal(2)
crashes. crash:	what to do when the system	crash(8)
binary, and/or manual for/	whereis: locate source,	whereis(1)
whodo:	who is doing what.	whodo(1M)
who:	who is on the system.	who(1)
	who: who is on the system.	who(1)
current user id.	whoami: print effective	whoami(1)
	whodo: who is doing what.	whodo(1M)
machines. rwho:	who's logged in on local	rwho(1N)
cd: change	working directory.	cd(1)
chdir: change	working directory.	chdir(2)
get pathname of current	working directory. getcwd:	getcwd(3C)
pwd:	working directory name.	pwd(1)
worm: Play the growing	worm game.	worm(6)
game.	worm: Play the growing worm	worm(6)
display terminal.	worms: animate worms on a	worms(6)
worms: animate	worms on a display terminal.	worms(6)
write:	write on a file.	write(3)
writev:	write on a file.	writev(3N)
putpwent:	write password file entry.	putpwent(3C)
wall:	write to all users.	wall(1M)
write:	write to another user.	write(1)
	write: write on a file.	write(3)
	write: write to another user.	write(1)
	writev: write on a file.	writev(3N)
file regions for reading or	writing. /provide exclusive	locking(2)
open: open for reading or	writing.	open(2)
utmp, wtmp: utmp and	wtmp entry formats.	utmp(4)
formats. utmp,	wtmp: utmp and wtmp entry	utmp(4)
accounting records. fwtmp,	wtmpfix: manipulate connect	fwtmp(1M)
hunt-the-wumpus.	wump: the game of	wump(6)
list(a) and execute command.	xargs: construct argument	xargs(1)

*Permuted Index*

Fortran bitwise/ and, or,	xor, not, lshift, rshift: . . . . .	bool(3F)
programs to implement shared/	xstr: extract strings from C . . . . .	xstr(1)
j0, j1, jn,	y0, y1, yn: Bessel functions. . . . .	bessel(3M)
j0, j1, jn, y0,	y1, yn: Bessel functions. . . . .	bessel(3M)
compiler-compiler.	yacc: yet another . . . . .	yacc(1)
j0, j1, jn, y0, y1,	yn: Bessel functions. . . . .	bessel(3M)
abs, iabs, dabs, cabs,	zabs: Fortran absolute value. . . . .	abs(3F)





**NAME**

intro — introduction to system maintenance commands and application programs

**DESCRIPTION**

This section describes, in alphabetical order, commands that are used chiefly for system maintenance and administration purposes. The commands in this section should be used along with those listed in Section 1 of the *User Manual*. References to other manual entries not of the form *name(1M)*, *name(7)* or *name(8)* refer to entries of that manual.

**COMMAND SYNTAX**

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

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

where:

<i>name</i>	The name of an executable file.
<i>option</i>	— <i>noargletter(s)</i> or, — <i>argletter&lt;&gt;optarg</i> where <> is optional white space.
<i>noargletter</i>	A single letter representing an option without an argument.
<i>argletter</i>	A single letter representing an option requiring an argument.
<i>optarg</i>	Argument (character string) satisfying preceding <i>argletter</i> .
<i>cmdarg</i>	Path name (or other command argument) <i>not</i> beginning with — or, — by itself indicating the standard input.

**SEE ALSO**

*getopt(1)*, *getopt(3C)*.

*User Manual*.

*Administrator Guide*.

**DIAGNOSTICS**

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

**BUGS**

Regretfully, many commands do not adhere to the aforementioned syntax.

**NAME**

accept, reject — allow/prevent LP requests

**SYNOPSIS**

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

**DESCRIPTION**

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

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

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

**FILES**

`/usr/spool/lp/*`

**SEE ALSO**

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

**NAME**

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

**SYNOPSIS**

```
/usr/lib/acct/acctdisk
/usr/lib/acct/acctdusg [-u file] [-p file]
/usr/lib/acct/accton [file]
/usr/lib/acct/acctwtmp "reason"
```

**DESCRIPTION**

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

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

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

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

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

*Acctdusg* reads its standard input (usually from *find / -print*) and computes disk resource consumption (including indirect blocks) by login. If *-u* is given, records consisting of those file names for which *acctdusg* charges no one are placed in *file* (a potential source for finding users trying to avoid disk charges). If *-p* is given, *file* is the name of the password file. This option is not needed if the password file is */etc/passwd*. (See *diskusg*(1M) for more details.)

*Accton* alone turns process accounting off. If *file* is given, it must be the name of an existing file, to which the kernel appends process accounting records (see *acct*(2) and *acct*(4)).

*Acctwtmp* writes a *utmp*(4) record to its standard output. The record contains the current time and a string of characters that describe the *reason*. A record type of ACCOUNTING is assigned (see *utmp*(4)). *Reason* must be a string of 11 or less characters, numbers, \$, or spaces. For example, the following are suggestions for use in reboot and shutdown procedures, respectively:

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

**FILES**

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

**SEE ALSO**

acctcms(1M), acctcom(1), acctcon(1M), acctmerg(1M), acctprc(1M), acctsh(1M), diskusg(1M), fwtmp(1M), runacct(1M), acct(2), acct(4), utmp(4).  
"ACCOUNTING" in the *Administrator Guide*.

**NAME**

acctcms — command summary from per-process accounting records

**SYNOPSIS**

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

**DESCRIPTION**

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

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

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

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

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

**EXAMPLE**

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

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

**SEE ALSO**

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

**BUGS**

Unpredictable output results if *-t* is used on new style internal summary format files, or if it is not used with old style internal summary format files.

**NAME**

acctcon1, acctcon2 — connect-time accounting

**SYNOPSIS**

/usr/lib/acct/acctcon1 [options]

/usr/lib/acct/acctcon2

**DESCRIPTION**

*Acctcon1* converts a sequence of login/logoff records read from its standard input to a sequence of records, one per login session. Its input should normally be redirected from */etc/wtmp*. Its output is ASCII, giving device, user ID, login name, prime connect time (seconds), non-prime connect time (seconds), session starting time (numeric), and starting date and time. The *options* are:

- p Print input only, showing line name, login name, and time (in both numeric and date/time formats).
- t *Acctcon1* maintains a list of lines on which users are logged in. When it reaches the end of its input, it emits a session record for each line that still appears to be active. It normally assumes that its input is a current file, so that it uses the current time as the ending time for each session still in progress. The -t flag causes it to use, instead, the last time found in its input, thus assuring reasonable and repeatable numbers for non-current files.
- l *file* *File* is created to contain a summary of line usage showing line name, number of minutes used, percentage of total elapsed time used, number of sessions charged, number of logins, and number of logoffs. This file helps track line usage, identify bad lines, and find software and hardware oddities. Hang-up, termination of *login(1)* and termination of the login shell each generate logoff records, so that the number of logoffs is often three to four times the number of sessions. See *init(1M)* and *utmp(4)*.
- o *file* *File* is filled with an overall record for the accounting period, giving starting time, ending time, number of reboots, and number of date changes.

*Acctcon2* expects as input a sequence of login session records and converts them into total accounting records (see *tacct* format in *acct(4)*).

**EXAMPLE**

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

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

**FILES**

*/etc/wtmp*

**SEE ALSO**

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

**BUGS**

The line usage report is confused by date changes. Use *wtmpfix* (see *fwtmp(1M)*) to correct this situation.

**ACCTCON1 (1M)**

**SEE *ACCTCON***

**ACCTCON1 (1M)**

**ACCTCON2 (1M)**

**SEE *ACCTCON***

**ACCTCON2 (1M)**

**ACCTDISK (1M)**

**SEE *ACCT***

**ACCTDISK (1M)**

**ACCTDUSG (1M)**

**SEE *ACCT***

**ACCTDUSG (1M)**

**NAME**

acctmerg - merge or add total accounting files

**SYNOPSIS**

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

**DESCRIPTION**

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

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

**EXAMPLE**

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

```
acctmerg -v <file1 >file2
          edit file2 as desired ...
acctmerg -i <file2 >file1
```

**SEE ALSO**

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



**ACCTON (1M)**

**SEE ACCT**

**ACCTON (1M)**

**NAME**

acctprc1, acctprc2 — process accounting

**SYNOPSIS**

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

`/usr/lib/acct/acctprc2`

**DESCRIPTION**

*Acctprc1* reads input in the form described by *acct(4)*, adds login names corresponding to user IDs, then writes for each process an ASCII line giving user ID, login name, prime CPU time (tics), non-prime CPU time (tics), and mean memory size (in memory segment units). If *ctmp* is given, it is expected to contain a list of login sessions, in the form described in *acctcon(1M)*, sorted by user ID and login name. If this file is not supplied, it obtains login names from the password file. The information in *ctmp* helps it distinguish among different login names that share the same user ID.

*Acctprc2* reads records in the form written by *acctprc1*, summarizes them by user ID and name, then writes the sorted summaries to the standard output as total accounting records.

**EXAMPLE**

These commands are typically used as shown below:

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

**FILES**

`/etc/passwd`

**SEE ALSO**

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

**BUGS**

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

**CAVEAT**

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

ACCTPRC1 (1M)

SEE ACCTPRC

ACCTPRC1 (1M)

ACCTPRC2 (1M)

SEE ACCTPRC

ACCTPRC2 (1M)

## NAME

chargefee, ckpacct, dodisk, lastlogin, monacct, nulladm, prttmp, prdaily, prtacct, runacct, shutacct, startup, turnacct – shell procedures for accounting

## SYNOPSIS

```

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

```

## DESCRIPTION

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

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

*Dodisk* should be invoked by *cron* to perform the disk accounting functions. By default, it will do disk accounting on the special files in */etc/checklist*. If the *-o* flag is used, it will do a slower version of disk accounting by login directory. *Files* specify the one or more filesystem names where disk accounting

will be done. If *files* are used, disk accounting will be done on these filesystems only. If the *-o* flag is used, *files* should be mount points of mounted filesystems. If omitted, they should be the special file names of mountable filesystems.

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

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

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

*Prctmp* can be used to print the session record file (normally */usr/adm/acct/nite/ctmp* created by *acctcon1* (see *acctcon*(1M)).

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

*Prtacct* can be used to format and print any total accounting (*tacct*) file.

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

*Shutacct* should be invoked during a system shutdown (usually in */etc/shutdown*) to turn process accounting off and append a "reason" record to */etc/wtmp*.

*Startup* should be called by */etc/rc* to turn the accounting on whenever the system is brought up.

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

## FILES

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

## SEE ALSO

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

ACCTWTMP(1M)

SEE ACCT

ACCTWTMP(1M)

**NAME**

**badblk** – program to set or update bad block information

**SYNOPSIS**

**badblk** [-w] [-mN] [-c] [-p] /dev/rXYZ [M]

**DESCRIPTION**

*Badblk* sets or updates bad block information for those disk drives that support soft sector bad block remapping.

If invoked with the **-w** option, write/verify is performed to determine if there is a bad block; otherwise only read is done.

If invoked with the **-c** option the current badblock is cleared.

If invoked with the **-p** option the current badblock table is printed.

If invoked with the **-mN** option, the number of alternate blocks will be set to N. *Badblk* returns an error if N > NICALT (currently 50).

/dev/rXYZ is the device name.

M is one or more block numbers separated by blanks.

If invoked with no specific block numbers and no bad block verification has been done before, then each block on the disk is checked (either read or write/verify) and bad block information in block 0 is set up from scratch.

If invoked with no specific block numbers, but block 0 already contains bad block information set up earlier, then a verification on the whole disk is performed; any new bad blocks not already on the block 0 table will be added.

If invoked with the device name plus block numbers, then only the indicated blocks are updated in block 0.

After alternate blocks are assigned, block 0 is updated and the updated blocks are verified to make sure alternate blocks are good. If alternate blocks are not good, new alternate block numbers are assigned.

A raw device that accesses the entire disk and allows for writing block zero should be specified.

**EXAMPLE**

**badblk -w /dev/rw1hw0**

does a full write/verify on winchester 1 and updates the header block. The rw1hw0 specifies raw (r) winchester 1 (w1), the full disk (h), with the capability of writing block 0 (w0).



**badblk /dev/rw1hw0 3754 8123**  
**adds blocks 3754 and 8123 to the badblock list.**

**BCHECKRC (1M)**

**SEE BRC**

**BCHECKRC (1M)**

**NAME**

bcopy - interactive block copy

**SYNOPSIS**

/etc/bcopy

**DESCRIPTION**

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

The following questions are asked:

- to:** (you name the file or device to be copied to).
- offset:** (you provide the starting "to" block number).
- from:** (you name the file or device to be copied from).
- offset:** (you provide the starting "from" block number).
- count:** (you reply with the number of blocks to be copied).

After **count** is exhausted, the **from** question is repeated (giving you a chance to concatenate blocks at the **to+offset+count** location). If you answer **from** with a carriage return, everything starts over.

Two consecutive carriage returns terminate *bcopy*.

**SEE ALSO**

cpio(1), dd(1).

**NAME**

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

**SYNOPSIS**

*/etc/brc*

*/etc/bcheckrc*

*/etc/rc*

*/etc/powerfail*

**DESCRIPTION**

Except for *powerfail*, these shell procedures are executed via entries in */etc/inittab* by *init(1M)* when the system is changed out of *SINGLE USER* mode. *Powerfail* is executed whenever a system power failure is detected.

The *brc* procedure clears the mounted file system table, */etc/mnttab* (see *mnttab(4)*), and loads any programmable micro-processors with their appropriate scripts.

The *bcheckrc* procedure performs all the necessary consistency checks to prepare the system to change into multi-user mode. It will prompt to set the system date and to check the file systems with *fsck(1M)*.

The *rc* procedure starts all system daemons before the terminal lines are enabled for multi-user mode. In addition, file systems are mounted and accounting, error logging, system activity logging and the Remote Job Entry (RJE) system are activated in this procedure.

The *powerfail* procedure is invoked when the system detects a power failure condition. Its chief duty is to reload any programmable micro-processors with their appropriate scripts, if suitable. It also logs the fact that a power failure occurred.

**SEE ALSO**

*fsck(1M)*, *init(1M)*, *shutdown(1M)*, *inittab(4)*, *mnttab(4)*.

**CHARGEFEE (1M)**

**SEE ACCTSH**

**CHARGEFEE (1M)**

**NAME**

checkall — faster file system checking procedure

**SYNOPSIS**

/etc/checkall

**DESCRIPTION**

The *checkall* procedure is a prototype and must be modified to suit local conditions. The following will serve as an example:

```
# check the root file system by itself
fsock /dev/dsk/0s0

# dual fsock of drives 0 and 1
dfsock /dev/rdisk/0s[12345] — /dev/rdisk/1s1
```

In the above example (where */dev/rdisk/1s1* is 320K blocks and */dev/rdisk/0s[12345]* are each 65K or less), a previous sequential *fsock* took 19 minutes. The *checkall* procedure takes 11 minutes.

*Dfsock* is a program that permits an operator to interact with two *fsock*(1M) programs at once. To aid in this, *dfsock* will print the file system name for each message to the operator. When answering a question from *dfsock*, the operator must prefix the response with a 1 or a 2 (indicating that the answer refers to the first or second file system group).

Due to the file system load balancing required for dual checking, the *dfsock*(1M) command should always be executed through the *checkall* shell procedure.

In a practical sense, the file systems are divided as follows:

```
dfsock file_systems_on_drive_0 — file_systems_on_drive_1
dfsock file_systems_on_drive_2 — file_systems_on_drive_3
. . .
```

A three-drive system can be handled by this more concrete example (assumes two large file systems per drive):

```
dfsock /dev/dsk/3s1 /dev/dsk/0s[14] — /dev/dsk/1s[14]
/dev/dsk/3s4
```

Note that the first file system on drive 3 is first in the *filesystems1* list and is last in the *filesystems2* list assuring that references to that drive will not overlap at execution time.

**WARNINGS**

1. Do not use *dfsock* to check the *root* file system.
2. On a check that requires a scratch file (see *-t* above), be careful not to use the same temporary file for the two groups (this is sure to scramble the file systems).
3. The *dfsock* procedure is useful only if the system is set up for multiple physical I/O buffers.

**SEE ALSO**

*dfsock*(1M), *fsock*(1M).

**NAME**

**chgnod** — change current UNIX system nodename

**SYNOPSIS**

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

**DESCRIPTION**

*Chgnod* accesses the structure defined in `<sys/utsname.h>`:

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

*Chgnod* changes the nodename of the currently running kernel to *new-name*. *Kernel-file* is the name of the kernel that was last booted. If no *kernel-file* is specified, */unix* is assumed. *Nodename* is a null terminated string containing the name that the system is known by on a communications network.

*New-name* must be no longer than eight characters; longer names are truncated to eight. The old and new nodenames are printed on completion.

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

**EXAMPLE**

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

will change your nodename to user10 if */unix.current* was the last unix booted.

**SEE ALSO**

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

**NAME**

chroot — change root directory for a command

**SYNOPSIS**

`/etc/chroot` newroot command

**DESCRIPTION**

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

Notice that:

```
chroot newroot command > x
```

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

This command is restricted to the *super-user*.

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

**EXAMPLE**

If you have a floppy-based UNIX system disk in `/dev/floppy`, then:

```
mkdir /t
mount /dev/flop /t
chroot /t /bin/sh
```

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

**SEE ALSO**

`chdir(2)`.

**BUGS**

One should exercise extreme caution when referencing special files in the new root file system.



CKPACCT (1M)

SEE ACCTSH

CKPACCT (1M)

**NAME**

config – configure system

**SYNOPSIS**

*/etc/config* [-t] [-cfile] [-hfile] [-lfile] [-mfile] *dfile*

**DESCRIPTION**

*Config* is a program that takes a description of a UNIX system and generates two files. One file provides information regarding the interface between the hardware and device handlers. The other file is a C program defining the configuration tables for the various devices on the system.

- t requests a short table of major device numbers for character and block type devices.
- c specifies the name of the configuration table file; *conf.c* is the default name.
- h specifies the name of the parameter header file; *config.h* is the default name.
- l specifies the name of the exception vector file; *ivec* is the default name.
- m specifies the name of the file that contains all the information regarding supported devices; */etc/master* is the default name. This file is supplied with Oreo and should not be modified unless the user fully understands its construction.

The description file *dfile* must contain device information for the user's system. This file is divided into three parts. The first part contains physical device specifications. The second part contains system-dependent information. The third part contains microprocessor-specific information. The first two parts are required, the third part is optional. Any line with an asterisk (\*) in column 1 is a comment.

**First Part of *dfile***

Each line contains four or five fields, delimited by blanks and/or tabs in the following format:

**devname**

the name of the device (as it appears in the *etc/master* device table).

**index**

passed to interrupt routine as channel id.

**address**  
the device address (hexadecimal)

**vector[:mod]**  
*vector* is the interrupt vector location (hexadecimal); *mod* is used to identify more than one interrupt routine.

There are certain drivers which may be provided with the system that are actually pseudo-device drivers, that is, there is no real hardware associated with the driver. Drivers of this type are identified on their respective manual entries. When these devices are specified in the description file, the interrupt *vector* device and *address* must be zero.

### Second Part of *dfile*

The second part contains three different types of lines. Note that all specifications of this part are required, although their order is arbitrary.

#### 1. *Root/pipe/dump device specification*

Four lines of at least three fields each:

```

root   devnameminor
pipe  devnameminor
dump  devnameminor

```

where *minor* is the minor device number (in decimal).

#### 2. *Swap device specification*

One line that contains five fields as follows:

```

swap  devnameminor swplo nswap

```

where *swplo* is the lowest disk block (decimal) in the swap area and *nswap* is the number of disk blocks (decimal) in the swap area.

#### 3. *Parameter specification*

Several lines of two fields each as follows (number is decimal). For example:

```

buffers   number
inodes    number
files     number
mounts    number
coremap   number

```

<b>swapmap</b>	number
<b>calls</b>	number
<b>procs</b>	number
<b>maxproc</b>	number
<b>texts</b>	number
<b>clists</b>	number
<b>hasbbuf</b>	number
<b>physbuf</b>	number
<b>power</b>	0 or 1
<b>mesg</b>	0 or 1
<b>sema</b>	0 or 1
<b>shmem</b>	0 or 1

**FILES**

<b>/etc/master</b>	default input master device table
<b>conf.c</b>	default output configuration table file
<b>config.h</b>	default output configuration header file

**SEE ALSO**

**sysdef(1M), master(4).**

**DIAGNOSTICS**

Diagnostics are routed to the standard output and are self-explanatory.

## NAME

config - configure system

## SYNOPSIS

/etc/config [ -t ] [ -cfile ] [ -hfile ] [ -lfile ] [ -mfile ] dfile

## DESCRIPTION

*Config* is a program that takes a description of a UNIX system and generates three files. One file provides information regarding the interface between the hardware and device handlers. The second file is a C program defining the configuration tables for the various devices on the system. The third is an include file with parameter defines (counts).

- t requests a short table of major device numbers for character and block type devices.
- c specifies the name of the configuration table file; *conf.c* is the default name.
- h specifies the name of the parameter header file; *config.h* is the default name.
- l specifies the name of the exception vector file; *ivec.s* is the default name.
- m specifies the name of the file that contains all the information regarding supported devices; */etc/master* is the default name. This file is supplied with UniPlus<sup>+</sup> and should not be modified unless the user fully understands its construction.

The description file *dfile*; must contain device information for the user's system. This file is divided into three parts. The first part contains physical device specifications. The second part contains system-dependent information. The third part contains microprocessor-specific information. The first two parts are required, the third part is optional. Any line with an asterisk (\*) in column 1 is a comment.

First Part of *dfile*

Each line contains four or five fields, delimited by blanks and/or tabs in the following format:

```
devname      the name of the device (as it appears in the etc/master device
              table).
index        passed to interrupt routine as channel id.
address      the device address (hexadecimal)
vector[:mod] vector is the interrupt vector location (decimal); mod is a character
              used to identify more than one interrupt routine, or more than
              one vector using the same interrupt routine.
```

There are certain drivers which may be provided with the system that are actually pseudo-device drivers, that is, there is no real hardware associated with the driver. Drivers of this type are identified on their respective manual entries. When these devices are specified in the description file, the interrupt *vector* device and *address* must be zero.

Second Part of *dfile*

The second part contains three different types of lines. Note that all specifications of this part are required, although their order is arbitrary.

1. *Root | pipe | dump device specification*

Four lines of at least three fields each:

```

root   devnameminor
pipe   devnameminor
dump   devnameminor

```

where *minor* is the minor device number (in decimal).

## 2. Swap device specification

One line that contains five fields as follows:

```

swap   devnameminor   swaplow   swapcnt

```

where *swaplow* is the lowest disk block (decimal) in the swap partition and *swapcnt* is the number of disk blocks (decimal) in the swap area.

## 3. Parameter specification

Several lines of two fields each as follows (number is decimal). For example:

```

buffers   number
inodes    number
files     number
mounts    number
coremap   number
swapmap   number
calls     number
procs     number
maxproc   number
texts     number
clists    number
hashbuf   number
physbuf   number
power     0 or 1
mesg      0 or 1
sema      0 or 1
shmema    0 or 1

```

## FILES

```

/etc/master      default input master device table
ivec.s           default output exception vector file for m68k
conf.c           default output configuration table file
config.h         default output configuration header file

```

## SEE ALSO

sysdef(1M), master(4).

## DIAGNOSTICS

Diagnostics are routed to the standard output and are self-explanatory.

**NAME**

*clri* - clear i-node

**SYNOPSIS**

*/etc/clri file-system i-number ...*

**DESCRIPTION**

*Clri* writes zeros on the 64 bytes occupied by the i-node numbered *i-number*. *File-system* must be a special file name referring to a device containing a file system. After *clri* is executed, any blocks in the affected file will show up as "missing" in an *fsck(1M)* of the *file-system*. This command should only be used in emergencies and extreme care should be exercised.

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

The primary purpose of this routine is to remove a file which for some reason appears in no directory. If it is used to *zap* an i-node which does appear in a directory, care should be taken to track down the entry and remove it. Otherwise, when the i-node is reallocated to some new file, the old entry will still point to that file. At that point removing the old entry will destroy the new file. The new entry will again point to an unallocated i-node, so the whole cycle is likely to be repeated again and again.

**EXAMPLE**

*clri /dev/yyyy n*

where "yyyy" is a legitimate system device name, and "n" is the inode number to be cleared, will cause inode numbered "n" for device "/dev/yyyy" to be cleared to 64-bytes of 0s. Note: this instruction should only be used with caution.

**SEE ALSO**

*fsck(1M)*, *fsdb(1M)*, *ncheck(1M)*, *fs(4)*.

**BUGS**

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

**NAME**

`cpset` - install object files in binary directories

**SYNOPSIS**

`cpset [-o] object directory [mode owner group]`

**DESCRIPTION**

`Cpset` is used to install the specified *object* file in the given *directory*. The *mode*, *owner*, and *group*, of the destination file may be specified on the command line. If this data is omitted, two results are possible:

If the user of `cpset` has administrative permissions (that is, the user's numerical ID is less than 100), the following defaults are provided:

mode - 0755  
owner - bin  
group - bin

If the user is not an administrator, the default, owner, and group of the destination file will be that of the invoker.

An optional argument of `-o` will force `cpset` to move *object* to **OLD***object* in the destination directory before installing the new object.

For example:

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

All the examples above have the same effect (assuming the user is an administrator). The file `echo` will be copied into `/bin` and will be given **0755**, **bin**, **bin** as the mode, owner, and group, respectively.

`Cpset` utilizes the file `/usr/src/destinations` to determine the final destination of a file. The locations file contains pairs of pathnames separated by spaces or tabs. The first name is the "official" destination (for example: `/bin/echo`). The second name is the new destination. For example, if `echo` is moved from `/bin` to `/usr/bin`, the entry in `/usr/src/destinations` would be:

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

When the actual installation happens, `cpset` verifies that the "old" pathname does not exist. If a file exists at that location, `cpset` issues a warning and continues. This file does not exist on a distribution tape; it is used by sites to track local command movement. The procedures used to build the source will be responsible for defining the "official" locations of the source.

**Cross Generation**

The environment variable `ROOT` will be used to locate the destination file (in the form `$ROOT/usr/src/destinations`). This is necessary in the cases where cross generation is being done on a production system.

**SEE ALSO**

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



**NAME**

*cron* - clock daemon

**SYNOPSIS**

*/etc/cron*

**DESCRIPTION**

*Cron* executes commands at specified dates and times. Regularly scheduled commands can be specified according to instructions found in *crontab* files; users can submit their own *crontab* file via the *crontab* command. Commands which are to be executed only once may be submitted via the *at* command. Since *cron* never exits, it should only be executed once. *Cron* is listed in the */etc/inittab* file and is therefore started directly by *init*(1M).

*Cron* examines *crontab* files and *at* command files only on certain occasions. This reduces the overhead of checking for new or changed files at regularly scheduled intervals. *Crontab* and *at* command files are checked only when one or more of the following conditions occurs:

- when the *cron* process is first initialized;
- when the system time is reset backward or forward;
- when a *crontab* file is changed using *crontab*(1)  
or when a new job is queued with *at*(1).

Note: submitting an *at* job causes the *at*(1) command file to be examined, but not the *crontab* files.

**FILES**

<i>/usr/lib/cron</i>	main <i>cron</i> directory
<i>/usr/lib/cron/log</i>	accounting information
<i>/usr/spool/cron</i>	spool area

**SEE ALSO**

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

**DIAGNOSTICS**

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

## NAME

*dcopy* - copy file systems for optimal access time

## SYNOPSIS

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

## DESCRIPTION

*Dcopy* copies file system *inputfs* to *outputfs*. *Inputfs* is the existing file system; *outputfs* is an appropriately sized file system, to hold the reorganized result. For best results *inputfs* should be the raw device and *outputfs* should be the block device. *Dcopy* should be run on unmounted file systems (in the case of the root file system, copy to a new pack). With no arguments, *dcopy* copies files from *inputfs* compressing directories by removing vacant entries, and spacing consecutive blocks in a file by the optimal rotational gap. The possible options are:

- sX supply device information for creating an optimal organization of blocks in a file. The forms of X are the same as the -s option of *fsck*(1M).
- an place the files not accessed in *n* days after the free blocks of the destination file system (default for *n* is 7). If no *n* is specified then no movement occurs.
- d leave order of directory entries as is (default is to move sub-directories to the beginning of directories).
- v currently reports how many files were processed, and how big the source and destination freelists are.
- x Debug mode. Prints out detailed information about blocks read and blocks examined.
- ffsize[:isize] specify the *outputfs* file system and inode list sizes (in blocks). If the option (or *:isize*) is not given, the values from the *inputfs* are used.

*Dcopy* catches interrupts and quits and reports on its progress. To terminate *dcopy*, send a quit signal and *dcopy* will no longer catch interrupts or quits.

## SEE ALSO

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

## NOTES

*Dcopy* now copies 512-block file systems and completely replaces the obsolete *dcopy1b*.

**NAME**

devnm — device name

**SYNOPSIS**

*/etc/devnm* [*names*]

**DESCRIPTION**

*Devnm* identifies the special file associated with the mounted file system where the argument *name* resides. (As a special case, both the block device name and the swap device name is printed for the argument */* if swapping is done on the same disk section as the root file system.) Argument names must be full path names.

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

**EXAMPLE**

```
/etc/devnm /usr
produces
/dev/dsk/0s1 /usr
if /usr is mounted on /dev/dsk/0s1.
```

**FILES**

*/dev/dsk/\**  
*/etc/mnttab*

**SEE ALSO**

*brc(1M)*, *setmnt(1M)*.

**NAME**

**df** - report number of free disk blocks

**SYNOPSIS**

**df** [ **-t** ] [ *file-systems* ]

**df** **-t** *file1* [ *file2* ] ...

**DESCRIPTION**

*Df* prints out the number of free blocks and free i-nodes available for on-line file systems by examining the counts kept in the super-blocks (the default system size for physical blocks is 512 bytes). *File-systems* may be specified either by device name (e.g., */dev/dsk1/0s1*) or by mounted directory name (e.g., */usr*). If the *file-systems* argument is unspecified, the free space on all of the mounted file systems is printed.

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

If the **-f** option is given, only an actual count of the blocks in the free list is made (free i-nodes are not reported). With this option, *df* will report on raw devices.

**FILES**

*/bin/df*

*/dev/dsk/\**

*/etc/mnttab*

**SEE ALSO**

*fs(4)*, *mnttab(4)*.

**DFSK (1M)**

**SEE FSCK**

**DFSK (1M)**

**NAME**

diskformat - format a disk

**SYNOPSIS**

diskformat [-size #] [-dens #] [-cyl f[-t]] [-sec f[-t]] [-i #] device

**DESCRIPTION**

*Diskformat* initializes a hard disk or floppy disk and formats it according to your specifications.

The following parameters may be specified ("device" is required):

- device** device to be formatted (must be raw device)
- size #** specify sector size in bytes
- dens #** specify density
- cyl #[-#]** format cylinders *f* to *t* (default *f*). A specification such as #- means "until the end".
- head #| -#|** Format heads *f* to *t* (default *f*). A specification such as #- means "until the end".
- sec #| -#|** Format sectors *f* to *t* (default *f*). A specification such as #- means "until the end".
- il #** Interleave factor for the disk.

**EXAMPLE**

```
diskformat /dev/rfdc0 -dens 1 -size 128 -il 3
```

will format the floppy disk on drive 0, single density, 128 bytes per sector with an interleave factor of 3. This format is the only truly portable floppy format.

**NAME**

disktune - tune floppy disk settling time parameters

**SYNOPSIS**

disktune [-srt #] [-hlt #] [-hut #] device

**DESCRIPTION**

*Disktune* tunes floppy disk settling time parameters. These include the motor stepping rate and the rate at which the head loads and unloads. *Disktune* thus enables you to obtain the most efficient operation from your floppy on those systems that support it.

If no settable parameters are given, *disktune* will report the current settings on *device*. *Disktune* retains the current settings on parameters which are not specified.

The raw device, */dev/rflop*, must be specified.

The settable parameters are:

- srt # seek motor stepping rate time in milliseconds
- hlt # head loading time in milliseconds
- hut # head unload time in milliseconds

**EXAMPLE**

disktune -srt 3 /dev/rfdc0

will set the step rate time on the floppy controller to 3 ms per step.

**BUGS**

Not supported on all drives.

**NAME**

diskusg — generate disk accounting data by user ID

**SYNOPSIS**

diskusg [options] [files]

**DESCRIPTION**

*Diskusg* generates intermediate disk accounting information from data in *files*, or the standard input if omitted. *Diskusg* output lines on the standard output, one per user, in the following format:

```
uid login # blocks
```

where

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

*Diskusg* normally reads only the i-nodes of file systems for disk accounting. In this case, *files* are the special filenames of these devices.

*Diskusg* recognizes the following options:

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

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

**EXAMPLES**

The following will generate daily disk accounting information:

```
for i in /dev/rp00 /dev/rp01 /dev/rp10 /dev/rp11; do
    diskusg $i > dtmp.`basename $i` &
done
wait
diskusg -s dtmp.* | sort +0n +1 | acctdisk > diskacct
```

**FILES**

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

**SEE ALSO**

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



**DODISK (1M)**

**SEE ACCTSH**

**DODISK (1M)**

**NAME**

**errdead** – extract error records from dump

**SYNOPSIS**

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

**DESCRIPTION**

When hardware errors are detected by the system, an error record that contains information pertinent to the error is generated. If the error-logging daemon *errdemon*(1M) is not active or if the system crashes before the record can be placed in the error file, the error information is held by the system in a local buffer. *Errdead* examines a system dump, extracts such error records, and passes them to *errpt*(1M) for analysis.

The *dumpfile* specifies the file (or memory) that is to be examined. The system *namelist* is specified by *namelist*; if not given, */unix* is used.

**FILES**

<i>/unix</i>	system <i>namelist</i>
<i>/usr/bin/errpt</i>	analysis program
<i>/usr/tmp/errXXXXXX</i>	temporary file

**DIAGNOSTICS**

Diagnostics may come from either *errdead* or *errpt*. In either case, they are intended to be self-explanatory.

**SEE ALSO**

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

**NAME**

*errdemon* — error-logging daemon

**SYNOPSIS**

*/usr/lib/errdemon* [*file*]

**DESCRIPTION**

The error logging daemon *errdemon* collects error records from the operating system by reading the special file */dev/error* and places them in *file*. If *file* is not specified when the daemon is activated, */usr/adm/errfile* is used. Note that *file* is created if it does not exist; otherwise, error records are appended to it, so that no previous error data is lost. No analysis of the error records is done by *errdemon*; that responsibility is left to *errpt*(1M). The error-logging daemon is terminated by sending it a software kill signal (see *kill*(1)). Only the super-user may start the daemon, and only one daemon may be active at any time.

**FILES**

*/dev/error*           source of error records  
*/usr/adm/errfile*   repository for error records

**DIAGNOSTICS**

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

**SEE ALSO**

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

## NAME

**errpt** - process a report of logged errors

## SYNOPSIS

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

## DESCRIPTION

*Errpt* processes data collected by the error logging mechanism (*errdemon*(1M)) and generates a report of that data. The default report is a summary of all errors posted in the files named. Options apply to all files and are described below. If no files are specified, *errpt* attempts to use */usr/adm/errfile*.

A summary report notes the options that may limit its completeness, records the time stamped on the earliest and latest errors encountered, and gives the total number of errors of one or more types. Each device summary contains the total number of unrecovered errors, recovered errors, errors unable to be logged, I/O operations on the device, and miscellaneous activities that occurred on the device. The number of times that *errpt* has difficulty reading input data is included as read errors.

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

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

- a** Produce a detailed report that includes all error types.
- dev** A detailed report is limited to data about devices given in *dev*, where *dev* can be one of two forms: a list of device identifiers separated from one another by a comma, or a list of device identifiers enclosed in double quotes and separated from one another by a comma and/or more spaces. *Errpt* is familiar with the common form of identifiers (see Section 7 of this volume). The devices for which errors are logged are system dependent. Additional identifiers are *int* and *mem* which include detailed reports of stray-interrupt and memory-parity type errors, respectively.
- e date** Ignore all records posted later than *date*, where *date* has the form *mmddhhmmyy*, consistent in meaning with the *date*(1) command.

- f** In a detailed report, limit the reporting of block device errors to unrecovered errors.
- p *n*** Limit the size of a detailed report to *n* pages.
- s *date*** Ignore all records posted earlier than *date*, where *date* has the form *mmddhhmmyy*, consistent in meaning with the *date(1)* command.

**FILES**

*/etc/master* for configuration of devices in system */usr/adm/errfile* default error file

**SEE ALSO**

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

**BUGS**

When illegal options are specified, *errpt* ignores them and generates default output.

**NAME**

errstop — terminate the error-logging daemon

**SYNOPSIS**

*/etc/errstop* [ *namelist* ]

**DESCRIPTION**

The error-logging daemon *errdemon(1M)* is terminated by using *errstop*. This is accomplished by executing *ps(1)* to determine the daemon's identity and then sending it a software kill signal (see *signal(2)*); */unix* is used as the system *namelist* if none is specified. Only the super-user may use *errstop*.

**FILES**

*/unix* default system *namelist*

**DIAGNOSTICS**

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

**SEE ALSO**

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

**NAME**

*ff* — list file names and statistics for a file system

**SYNOPSIS**

*/etc/ff* [*options*] *special*

**DESCRIPTION**

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

path-name i-number

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

path-name i-number size uid

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

- I Do not print the i-node number after each path name.
- l Generate a supplementary list of all path names for multiply linked files.
- p *prefix* The specified *prefix* will be added to each generated path name. The default is ..
- s Print the file size, in bytes, after each path name.
- u Print the owner's login name after each path name.
- a *n* Select if the i-node has been accessed in *n* days.
- m *n* Select if the i-node has been modified in *n* days.
- c *n* Select if the i-node has been changed in *n* days.
- n *file* Select if the i-node has been modified more recently than the argument *file*.
- i *i-node-list*

Generate names for only those i-nodes specified in *i-node-list*.

**EXAMPLE**

*ff -I /dev/diskroot*

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

*ff -m -l /dev/diskusr > /log/incbackup/usr/tuesday*

produces an index of files and i-numbers which are on a file system and have been modified in the last 24 hours.

*ff -i 451,76 /dev/rdisk/0s7*

obtains the path names for i-nodes 451 and 76 on a specified file system.

**SEE ALSO**

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

**BUGS**

Only a single path name out of any possible ones will be generated for a

**FF(1M)**

**FF(1M)**

multiply linked i-node, unless the **-l** option is specified. When **-l** is specified, no selection criteria apply to the names generated. All possible names for every linked file on the file system will be included in the output. On very large file systems, memory may run out before *ff* does.



**NAME**

filesave, tapesave — daily/weekly UNIX file system backup

**SYNOPSIS**

/etc/filesave.?

/etc/tapesave

**DESCRIPTION**

These shell scripts are provided as models. They are designed to provide a simple, interactive operator environment for file backup. *Filesave.?* is for daily disk-to-disk backup and *tapesave* is for weekly disk-to-tape.

The suffix *.?* can be used to name another system where two (or more) machines share disk drives (or tape drives) and one or the other of the systems is used to perform backup on both.

**SEE ALSO**

shutdown(1M), volcopy(1M).

**NAME**

*finc* — fast incremental backup

**SYNOPSIS**

*finc* [*selection-criteria*] *file-system raw-tape*

**DESCRIPTION**

*Finc* selectively copies the input *file-system* to the output *raw-tape*. The cautious will want to mount the input *file-system* read-only to insure an accurate backup, although acceptable results can be obtained in read-write mode. The tape must be previously labelled by *labelit* (see *volcopy*(1M)). The selection is controlled by the *selection-criteria*, accepting only those inodes/files for whom the conditions are true.

It is recommended that production of a *finc* tape be preceded by the *ff* command, and the output of *ff* be saved as an index of the tape's contents. Files on a *finc* tape may be recovered with the *frec* command.

The argument *n* in the *selection-criteria* which follow is used as a decimal integer (optionally signed), where *+n* means more than *n*, *-n* means less than *n*, and *n* means exactly *n*. A day is defined as a 24 hours.

- a n* True if the file has been accessed in *n* days.
- m n* True if the file has been modified in *n* days.
- c n* True if the i-node has been changed in *n* days.
- n file* True for any file which has been modified more recently than the argument *file*.

**EXAMPLE**

```
finc -m -2 /dev/rdiskusr /dev/rmt/0m
```

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

**SEE ALSO**

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

**NAME**

*freq* — recover files from a backup tape

**SYNOPSIS**

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

**DESCRIPTION**

*Frec* recovers files from the specified *raw-tape* backup tape written by *volcopy(1M)* or *finc(1M)*, given their *i-numbers*. The data for each recovery request will be written into the file given by *name*.

The *-p path* option allows you to specify a default prefixing *path* different from your current working directory. This will be prefixed to any *names* that are not fully qualified, i.e., that do not begin with */* or *./*. If any directories are missing in the paths of recovery *names* they will be created.

*-p path* Specifies a prefixing *path* to be used to fully qualify any names that do not start with */* or *./*.

*-f reqfile* Specifies a file which contains recovery requests. The format is *i-number:newname*, one per line.

**EXAMPLE**

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

recovers a file, *i-number* 1216 when backed-up, into a file named **junk** in your current working directory.

```
freq -p /usr/src/cmd /dev/rmt/0m 14156:a 1232:b
3141:/usr/joe/a.c
```

recovers files with *i-numbers* 14156, 1232, and 3141 into files */usr/src/cmd/a*, */usr/src/cmd/b* and */usr/joe/a.c*.

**SEE ALSO**

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

**BUGS**

While paving a path (i.e., creating the intermediate directories contained in a *pathname*) *freq* can only recover inode fields for those directories contained on the tape and requested for recovery.

## NAME

*fsck*, *dfsck* – file system consistency check and interactive repair

## SYNOPSIS

*/etc/fsck* [-y] [-n] [-sX] [-SX] [-t file] [-q] [-D] [-f] [file-systems]

*/etc/dfsck* [ options1 ] filesystem ... - [ options2 ] filesystem ...

## DESCRIPTION

*Fsck*

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

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

The following options are interpreted by *fsck*.

- y Assume a yes response to all questions asked by *fsck*.
- n Assume a no response to all questions asked by *fsck*; do not open the file system for writing.
- sX Ignore the actual free list and (unconditionally) reconstruct a new one by rewriting the super-block of the file system. The file system should be unmounted while this is done; if this is not possible, care should be taken that the system is quiescent and that it is rebooted immediately afterwards. This precaution is necessary so that the old, bad, in-core copy of the superblock will not continue to be used, or written on the file system.

The *-sX* option allows for creating an optimal free-list organization. The following forms of *X* are supported for the following devices:

- s3 (RP03)
- s4 (RP04, RP05, RP06)
- sBlocks-per-cylinder:Blocks-to-skip (for anything else)

If *X* is not given, the values used when the file system was created are used. If these values were not specified, then the value 400:7 is used.

- SX Conditionally reconstruct the free list. This option is like -sX above except that the free list is rebuilt only if there were no discrepancies discovered in the file system. Using -S will force a no response to all questions asked by *fsck*. This option is useful for forcing free list reorganization on uncontaminated file systems.
- t If *fsck* cannot obtain enough memory to keep its tables, it uses a scratch file. If the -t option is specified, the file named in the next argument is used as the scratch file, if needed. Without the -t flag, *fsck* will prompt the operator for the name of the scratch file. The file chosen should not be on the file system being checked, and if it is not a special file or did not already exist, it is removed when *fsck* completes.
- q Quiet *fsck*. Do not print size-check messages in Phase 1. Unreferenced fifos will silently be removed. If *fsck* requires it, counts in the superblock will be automatically fixed and the free list salvaged.
- D Directories are checked for bad blocks. Useful after system crashes.
- f Fast check. Check block and sizes (Phase 1) and check the free list (Phase 5). The free list will be reconstructed (Phase 6) if it is necessary.

If no *file-systems* are specified, *fsck* will read a list of default file systems from the file */etc/checklist*.

Inconsistencies checked are as follows:

1. Blocks claimed by more than one inode or the free list.
2. Blocks claimed by an inode or the free list outside the range of the file system.
3. Incorrect link counts.
4. Size checks:
  - Incorrect number of blocks.
  - Directory size not 16-byte aligned.
5. Bad inode format.
6. Blocks not accounted for anywhere.
7. Directory checks:
  - File pointing to unallocated inode.
  - Inode number out of range.
8. Super Block checks:

More than 65536 inodes.

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

9. Bad free block list format.
10. Total free block and/or free inode count incorrect.

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

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

#### **Fsck1b**

*Fsck* now performs file checks on 512-block file systems. *Fsck* completely replaces the obsolete *fsck1b* command.

#### **Dfsck**

*Dfsck* allows two file system checks on two different drives simultaneously. *options1* and *options2* are used to pass options to *fsck* for the two sets of file systems. A - is the separator between the file system groups.

The *dfsck* program permits an operator to interact with two *fsck*(1M) programs at once. To aid in this, *dfsck* will print the file system name for each message to the operator. When answering a question from *dfsck*, the operator must prefix the response with a 1 or a 2 (indicating that the answer refers to the first or second file system group).

Do not use *dfsck* to check the *root* file system.

#### **EXAMPLE**

```
fsck /dev/rdisk0
```

checks the consistency of device *rdisk0*.

#### **FILES**

*/etc/checklist* contains default list of file systems to check.

#### **SEE ALSO**

*crl*(1M), *ncheck*(1M), *checklist*(4), *fs*(4), *crash*(8).

**BUGS**

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

**DIAGNOSTICS**

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

**NAME**

*fscv* — convert files between M68000 and VAX-11/780 processors

**SYNOPSIS**

```
/etc/fscv -v ispecial [ ospecial ]
/etc/fscv -m ispecial [ ospecial ]
```

**DESCRIPTION**

*Fscv* converts file systems between M68000 and VAX-11/780 formats. The super block, free list, and inodes are converted to the format of the output file. *Fscv* may be executed on M68000 and VAX processors. The mandatory flag specifies the format of the converted file system:

- v Convert file system from M68000 to VAX format.
- m Convert file system from VAX to M68000 format.

*ispecial* is the name of a special file containing a file system to be converted (e.g., */dev/rdk10*). The optional *ospecial* is the name of the special file to receive the results of the conversion. If *ospecial* is specified, the entire contents of *ispecial* are copied to *ospecial* before the conversion is performed. If *ospecial* is not specified, an in-place conversion of *ispecial* is performed. The following items should be noted before executing *fscv*:

1. A file system consistency check (*fsck(1M)*) should be performed on *ispecial* immediately prior to executing *fscv*.
2. Neither *ispecial* nor the optional *ospecial* should contain a mounted file system during execution of *fscv*. Modification to either the input or the output file system while *fscv* is executing will probably corrupt the converted file system.
3. A backup of *ispecial* (see *volcopy(1M)*) is highly recommended if an in-place conversion is to be performed. System crashes, I/O errors, etc., during execution of *fscv* may destroy the file system contained in *ispecial*. Also, if the optional *ospecial* is specified, any data contained in that special file is over written.
4. If the optional *ospecial* is specified, this special file must be large enough to contain the entire contents of *ispecial*. See the appropriate special files in section 4.

**EXAMPLES**

Copy and convert a file system from M68000 to VAX format:

```
/etc/fscv -v /dev/rdk00 /dev/rdk10
```

Perform an in-place conversion from VAX to M68000 format:

```
/etc/fscv -m /dev/rdk10
```

**BUGS**

The boot block is not modified during conversion; the resulting file system is not bootable. No data contained in the files of the file system are modified.

**SEE ALSO**

*fsck(1M)*, *volcopy(1M)*.



**NAME**

*fsdb* — file system debugger

**SYNOPSIS**

*/etc/fsdb* special [ - ]

*/etc/fsdb1b* special [ - ]

**DESCRIPTION**

*Fsdb* can be used to patch up a damaged file system after a crash. It has conversions to translate block and i-numbers into their corresponding disk addresses. Also included are mnemonic offsets to access different parts of an i-node. These greatly simplify the process of correcting control block entries or descending the file system tree. *Fsdb1b* is a file system debugger for 512-block file systems. *Fsdb* automatically calls *fsdb1b* on 512-block file systems, but you can also call *fsdb1b* directly.

*Fsdb* contains several error checking routines to verify i-node and block addresses. These can be disabled if necessary by invoking *fsdb* with the optional - argument or by the use of the O symbol. (*Fsdb* reads the i-size and f-size entries from the superblock of the file system as the basis for these checks.)

Numbers are considered decimal by default. Octal numbers must be prefixed with a zero. During any assignment operation, numbers are checked for a possible truncation error due to a size mismatch between source and destination.

*Fsdb* reads a block at a time and will therefore work with raw as well as block I/O. A buffer management routine is used to retain commonly used blocks of data in order to reduce the number of read system calls. All assignment operations result in an immediate write-through of the corresponding block.

The symbols recognized by *fsdb* are:

#	absolute address
i	convert from i-number to i-node address
b	convert to block address
d	directory slot offset
+, -	address arithmetic
q	quit
>, <	save, restore an address
=	numerical assignment
= +	incremental assignment
= -	decremental assignment
" , ' , *	character string assignment
O	error checking flip flop
p	general print facilities
f	file print facility
B	byte mode
W	word mode
D	double word mode
!	escape to shell

The print facilities generate a formatted output in various styles. The current address is normalized to an appropriate boundary before printing begins. It advances with the printing and is left at the address of the last item printed. The output can be terminated at any time by typing the

delete character. If a number follows the **p** symbol, that many entries are printed. A check is made to detect block boundary overflows since logically sequential blocks are generally not physically sequential. If a count of zero is used, all entries to the end of the current block are printed. The print options available are:

<b>i</b>	print as i-nodes
<b>d</b>	print as directories
<b>o</b>	print as octal words
<b>e</b>	print as decimal words
<b>c</b>	print as characters
<b>b</b>	print as octal bytes

The **f** symbol is used to print data blocks associated with the current i-node. If followed by a number, that block of the file is printed. (Blocks are numbered from zero.) The desired print option letter follows the block number, if present, or the **f** symbol. This print facility works for small as well as large files. It checks for special devices and that the block pointers used to find the data are not zero.

Dots, tabs and spaces may be used as function delimiters but are not necessary. A line with just a new-line character will increment the current address by the size of the data type last printed. That is, the address is set to the next byte, word, double word, directory entry or i-node, allowing the user to step through a region of a file system. Information is printed in a format appropriate to the data type. Bytes, words and double words are displayed with the octal address followed by the value in octal and decimal. A **.B** or **.D** is appended to the address for byte and double word values, respectively. Directories are printed as a directory slot offset followed by the decimal i-number and the character representation of the entry name. Inodes are printed with labeled fields describing each element.

The following mnemonics are used for i-node examination and refer to the current working i-node:

<b>md</b>	mode
<b>ln</b>	link count
<b>uid</b>	user ID number
<b>gid</b>	group ID number
<b>sz</b>	file size
<b>a#</b>	data block numbers (0 - 12)
<b>at</b>	access time
<b>mt</b>	modification time
<b>maj</b>	major device number
<b>min</b>	minor device number

#### EXAMPLE

**386i** prints i-number 386 in an i-node format. This now becomes the current working i-node.

**ln=4** changes the link count for the working i-node to 4.

**ln=+1** increments the link count by 1.

**fc** prints, in ASCII, block zero of the file associated with the working i-node.

**2i.fd** prints the first 32 directory entries for the root i-node of this file system.

d5i.fc changes the current i-node to that associated with the 5th directory entry (numbered from zero) found from the above command. The first logical block of the file is then printed in ASCII.

512B.p0o

prints the superblock of this file system in octal.

2i.a0b.d7=3

changes the i-number for the seventh directory slot in the root directory to 3. This example also shows how several operations can be combined on one command line.

d7.nm="name"

changes the name field in the directory slot to the given string. Quotes are optional when used with nm if the first character is alphabetic.

a2b.p0d

prints the third block of the current inode as directory entries.

**SEE ALSO**

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

**NAME**

fuser - identify processes using a file or file structure

**SYNOPSIS**

`/etc/fuser [-ku] files [-] [[-ku] files] [-nnamelist]`

**DESCRIPTION**

*Fuser* lists the process IDs of the processes using the *files* specified as arguments. For block special devices, all processes using any file on that device are listed. The process ID is followed by *c*, *p* or *r* if the process is using the file as its current directory, the parent of its current directory (only when in use by the system), or its root directory, respectively. If the `-u` option is specified, the login name, in parentheses, also follows the process ID. In addition, if the `-k` option is specified, the SIGKILL signal is sent to each process. Only the super-user can terminate another user's process (see *kill(2)*). Options may be respecified between groups of files. The new set of options replaces the old set, with a lone dash canceling any options currently in force.

The process IDs are printed as a single line on the standard output, separated by spaces and terminated with a single new line. All other output is written on standard error. The `-n` option specifies an alternate namelist (`/unix` is the default).

**EXAMPLE**

`fuser -ku /dev/dsk/ls?`

will terminate all processes that are preventing disk drive one from being unmounted if typed by the super-user, listing the process ID and login name of each as it is killed.

`fuser -u /etc/passwd`

will list process IDs and login names of processes that have the password file open.

`fuser -ku /dev/dsk/ls? -u /etc/passwd`

will do both of the above examples in a single command line.

**FILES**

<code>/unix</code>	for namelist
<code>/dev/kmem</code>	for system image
<code>/dev/mem</code>	also for system image
<code>/dev/swap</code>	for outswapped processes

**SEE ALSO**

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

**NAME**

*fwtmp*, *wtmpfix* -- manipulate connect accounting records

**SYNOPSIS**

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

**DESCRIPTION****Fwtmp**

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

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

**Wtmpfix**

*Wtmpfix* examines the standard input or named files in *wtmp* format, corrects the time/date stamps to make the entries consistent, and writes to the standard output. A *-* can be used in place of *files* to indicate the standard input. If time/date corrections are not performed, *acctcon1* will fault when it encounters certain date-change records.

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

In addition to correcting time/date stamps, *wtmpfix* will check the validity of the name field to ensure that it consists solely of alphanumeric characters or spaces. If it encounters a name that is considered invalid, it will change the login name to *INVALID* and write a diagnostic to the standard error. In this way, *wtmpfix* reduces the chance that *acctcon1* will fail when processing connect accounting records.

**FILES**

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

**SEE ALSO**

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

**NAME**

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

**SYNOPSIS**

```
/etc/getty [ -h ] [ -t timeout ] line [ speed [ type [ linedisc ] ] ]
/etc/getty -c file
```

**DESCRIPTION**

*Getty* is a program that is invoked by *init*(1M). It is the second process in the series, (*init-getty-login-shell*) that ultimately connects a user with the UNIX system. Initially *getty* generates a system identification message from the values returned by the *uname*(2) system call. Then, if */etc/issue* exists, it outputs this to the user's terminal, followed finally by the login message field for the entry it is using from */etc/gettydefs*. *Getty* reads the user's login name and invokes the *login*(1) command with the user's name as argument. While reading the name, *getty* attempts to adapt the system to the speed and type of terminal being used.

*Line* is the name of a tty line in */dev* to which *getty* is to attach itself. *Getty* uses this string as the name of a file in the */dev* directory to open for reading and writing. Unless *getty* is invoked with the *-h* flag, *getty* will force a hangup on the line by setting the speed to zero before setting the speed to the default or specified speed. The *-t* flag plus *timeout* in seconds, specifies that *getty* should exit if the open on the line succeeds and no one types anything in the specified number of seconds. The optional second argument, *speed*, is a label to a speed and tty definition in the file */etc/gettydefs*. This definition tells *getty* at what speed to initially run, what the login message should look like, what the initial tty settings are, and what speed to try next should the user indicate that the speed is inappropriate (by typing a *<break>* character). The default *speed* is 300 baud. The optional third argument, *type*, is a character string describing to *getty* what type of terminal is connected to the line in question. *Getty* understands the following types:

<b>none</b>	default
<b>vt61</b>	DEC vt61
<b>vt100</b>	DEC vt100
<b>hp45</b>	Hewlett-Packard HP45
<b>c100</b>	Concept 100

The default terminal is *none*; i.e., any crt or normal terminal unknown to the system. Also, for terminal type to have any meaning, the virtual terminal handlers must be compiled into the operating system. They are available, but not compiled in the default condition. The optional fourth argument, *linedisc*, is a character string describing which line discipline to use in communicating with the terminal. Again the hooks for line disciplines are available in the operating system but there is only one presently available, the default line discipline, *LDISC0*.

When given no optional arguments, *getty* sets the *speed* of the interface to 300 baud, specifies that raw mode is to be used (awaken on every character), that echo is to be suppressed, either parity allowed, new-line characters will be converted to carriage return-line feed, and tab expansion performed on the standard output. It types the login message before reading the user's name a character at a time. If a null character (or framing error) is received, it is assumed to be the result of the user pushing the "break" key. This will cause *getty* to attempt the next *speed* in the series. The series

that *getty* tries is determined by what it finds in */etc/gettydefs*.

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

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

In addition to the standard UNIX system erase and kill characters (# and @), *getty* also understands \b as an erase, or ^U as a kill character, *getty* sets the standard erase character and/or kill character to match.

*Getty* also understands the "standard" ESS2 protocols for erasing, killing and aborting a line, and terminating a line. If *getty* sees the ESS erase character, \_, or kill character, \$, or abort character, &, or the ESS line terminators, / or !, it arranges for this set of characters to be used for these functions.

Finally, *login* is called with the user's name as an argument. Additional arguments may be typed after the login name. These are passed to *login*, which will place them in the environment (see *login(1)*).

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

## FILES

*/etc/gettydefs*  
*/etc/issue*

## SEE ALSO

*ct(1C)*, *init(1M)*, *login(1)*, *ioctl(2)*, *gettydefs(4)*, *inittab(4)*, *tty(7)*.

## BUGS

While *getty* does understand simple single character quoting conventions, it is not possible to quote the special control characters that *getty* uses to determine when the end of the line has been reached, which protocol is being used, and what the erase character is. Therefore, it is not possible to login via *getty* and type a #, @, /, !, \_, backspace, ^U, ^D, or & as part of your login name or argument. They will always be interpreted as having their special meaning as described above.

**GRPCK (1M)**

**SEE PWCK**

**GRPCK (1M)**



## NAME

init, telinit — process control initialization

## SYNOPSIS

```
/etc/init [0123456SsQq]
/etc/telinit [0123456sSQqabc]
```

## DESCRIPTION

**Init**

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

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

*Init* is invoked inside the UNIX system as the last step in the boot procedure. The first thing *init* does is to look for `/etc/inittab` and see if there is an entry of the type *initdefault* (see `inittab(4)`). If there is, *init* uses the *run-level* specified in that entry as the initial *run-level* to enter. If this entry is not in *inittab* or *inittab* is not found, *init* requests that the user enter a *run-level* from the virtual system console, `/dev/syscon`. If an S (s) is entered, *init* goes into the *SINGLE USER* level. This is the only *run-level* that doesn't require the existence of a properly formatted *inittab* file. If `/etc/inittab` doesn't exist, then by default the only legal *run-level* that *init* can enter is the *SINGLE USER* level. In the *SINGLE USER* level the virtual console terminal `/dev/syscon` is opened for reading and writing and the command `/bin/su` is invoked immediately. To exit from the *SINGLE USER* *run-level* one of two options can be elected. First, if the shell is terminated (via an end-of-file), *init* will reprompt for a new *run-level*. Second, the *init* or *telinit* command can signal *init* and force it to change the *run-level* of the system.

When attempting to boot the system, failure of *init* to prompt for a new *run-level* may be due to the fact that the device `/dev/syscon` is linked to a device other than the physical system teletype (`/dev/systty`). If this occurs, *init* can be forced to relink `/dev/syscon` by typing a delete on the system teletype which is co-located with the processor.

When *init* prompts for the new *run-level*, the operator may enter only one of the digits 0 through 6 or the letters S or s. If S is entered *init* operates as previously described in *SINGLE USER* mode with the additional result that `/dev/syscon` is linked to the user's terminal line, thus making it the virtual system console. A message is generated on the physical console, `/dev/systty`, saying where the virtual terminal has been relocated.

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

written by *init* whenever *SINGLE USER* mode is entered. If this file does not exist when *init* wants to read it, a warning is printed and default settings are assumed.

If a 0 through 6 is entered *init* enters the corresponding *run-level*. Any other input will be rejected and the user will be re-prompted. If this is the first time *init* has entered a *run-level* other than *SINGLE USER*, *init* first scans *inittab* for special entries of the type *boot* and *bootwait*. These entries are performed, providing the *run-level* entered matches that of the entry before any normal processing of *inittab* takes place. In this way any special initialization of the operating system, such as mounting file systems, can take place before users are allowed onto the system. The *inittab* file is scanned to find all entries that are to be processed for that *run-level*.

*Run-level 2* is usually defined by the user to contain all of the terminal processes and daemons that are spawned in the multi-user environment.

In a multi-user environment, the *inittab* file is usually set up so that *init* will create a process for each terminal on the system.

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

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

If *init* receives a *powerfail* signal (*SIGPWR*) and is not in *SINGLE USER* mode, it scans *inittab* for special powerfail entries. These entries are invoked (if the *run-levels* permit) before any further processing takes place. In this way *init* can perform various cleanup and recording functions whenever the operating system experiences a power failure.

When *init* is requested to change *run-levels* (via *telinit*), *init* sends the warning signal (*SIGTERM*) to all processes that are undefined in the target *run-level*. *Init* waits 20 seconds before forcibly terminating these processes via the kill signal (*SIGKILL*).

### Telinit

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

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

**a,b,c**

tells *init* to process only those */etc/inittab* file entries having the **a**, **b** or **c** *run-level* set.

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

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

*Telinit* can only be run by someone who is super-user or a member of group *sys*.

#### FILES

*/etc/inittab*  
*/etc/utmp*  
*/etc/wtmp*  
*/etc/ioctl.syscon*  
*/dev/syscon*  
*/dev/systty*

#### SEE ALSO

*getty*(1M), *login*(1), *sh*(1), *who*(1), *kill*(2), *inittab*(4), *utmp*(4).

#### DIAGNOSTICS

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

## NAME

install — install commands

## SYNOPSIS

/etc/install [-c *dira*] [-f *dirb*] [-i] [-n *dirc*] [-o] [-s] *file* [*dirx* ...]

## DESCRIPTION

*Install* is a command most commonly used in "makefiles" (see *make(1)*) to install a *file* (updated target file) in a specific place within a file system. Each *file* is installed by copying it into the appropriate directory, thereby retaining the mode and owner of the original command. The program prints messages telling the user exactly what files it is replacing or creating and where they are going.

If no options or directories (*dirx* ...) are given, *install* will search a set of default directories (*/bin*, */usr/bin*, */etc*, */lib*, and */usr/lib*, in that order) for a file with the same name as *file*. When the first occurrence is found, *install* issues a message saying that it is overwriting that file with *file*, and proceeds to do so. If the file is not found, the program states this and exits without further action.

If one or more directories (*dirx* ...) are specified after *file*, those directories will be searched before the directories specified in the default list.

The meanings of the options are:

- c *dira* Installs a new command (*file*) in the directory specified by *dira*, only if it is not found. If it is found, *install* issues a message saying that the file already exists, and exits without overwriting it. May be used alone or with the -s option.
- f *dirb* Forces *file* to be installed in given directory, whether or not one already exists. If the file being installed does not already exist, the mode and owner of the new file will be set to 755 and *bin*, respectively. If the file already exists, the mode and owner will be that of the already existing file. May be used alone or with the -o or -s options.
- i Ignores default directory list, searching only through the given directories (*dirx* ...). May be used alone or with any other options other than -c and -f.
- n *dirc* If *file* is not found in any of the searched directories, it is put in the directory specified in *dirc*. The mode and owner of the new file will be set to 755 and *bin*, respectively. May be used alone or with any other options other than -c and -f.
- o If *file* is found, this option saves the "found" file by copying it to *OLDfile* in the directory in which it was found. This option is useful when installing a normally text busy file such as */bin/sh* or */etc/getty*, where the existing file cannot be removed. May be used alone or with any other options other than -c.
- s Suppresses printing of messages other than error messages. May be used alone or with any other options.

## SEE ALSO

cpset(1M), make(1).

**NAME**

killall — kill all active processes

**SYNOPSIS**

*/etc/killall* [*-n*namelist] [signal]

**DESCRIPTION**

*Killall* is a procedure used by */etc/shutdown* to kill all active processes not directly related to the shutdown procedure.

If you use the *-n*namelist option, the argument will be taken as the name of an alternate *namelist* file in place of */unix*.

*Killall* is chiefly used to terminate all processes with open files so that the mounted file systems will be unbusied and can be unmounted.

*Killall* sends *signal* (see *kill(1)*) to all remaining processes not belonging to the above group of exclusions. If no *signal* is specified, a default of 9 is used.

**FILES**

*/etc/shutdown*

**SEE ALSO**

*fuser(1M)*, *kill(1)*, *ps(1)*, *shutdown(1M)*, *signal(2)*.

**LABELIT (1M)**

**SEE *VOLCOPY***

**LABELIT (1M)**

**LASTLOGIN (1M)**

**SEE *ACCTSH***

**LASTLOGIN (1M)**

**NAME**

link, unlink — exercise link and unlink system calls

**SYNOPSIS**

```
/etc/link file1 file2
/etc/unlink file
```

**DESCRIPTION**

*Link* and *unlink* perform their respective system calls on their arguments, abandoning all error checking. These commands may only be executed by the super-user, who (it is hoped) knows what he or she is doing.

**EXAMPLE**

```
link file1 file2
```

creates a directory entry for "file2" with the same inode number as "file1".  
NOTE: *link* should be used with extreme caution.

**SEE ALSO**

rm(1), link(2), unlink(2).

## NAME

lpadmin — configure the LP spooling system

## SYNOPSIS

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

## DESCRIPTION

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

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

- d [dest]* makes *dest*, an existing destination, the new system default destination. If *dest* is not supplied, then there is no system default destination. This option may be used when *lpsched*(1M) is running. No other *options* are allowed with *-d*.
- x dest* removes destination *dest* from the LP system. If *dest* is a printer and is the only member of a class, then the class will be deleted, too. No other *options* are allowed with *-x*.
- p printer* names a *printer* to which all of the *options* below refer. If *printer* does not exist then it will be created.

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

- c class* inserts printer *P* into the specified *class*. *Class* will be created if it does not already exist.
- e printer* copies an existing *printer's* interface program to be the new interface program for *P*.
- h* indicates that the device associated with *P* is hardwired. This *option* is assumed when creating a new printer unless the *-l* *option* is supplied.
- i interface* establishes a new interface program for *P*. *Interface* is the pathname of the new program.
- l* indicates that the device associated with *P* is a login terminal. The LP scheduler, *lpsched*, disables all login terminals automatically each time it is started. Before re-enabling *P*, its current *device* should be established using *lpadmin*.
- m model* selects a model interface program for *P*. *Model* is one of the model interface names supplied with the LP software (see *Models* below).
- r class* removes printer *P* from the specified *class*. If *P* is the last member of the *class*, then the *class* will be removed.
- v device* associates a new *device* with printer *P*. *Device* is the pathname of a file that is writable by the LP administrator, *lp*. Note that there is nothing to stop an administrator from



associating the same *device* with more than one *printer*. If only the `-p` and `-v` options are supplied, then *lpadmin* may be used while the scheduler is running.

### Restrictions.

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

### Models.

Model printer interface programs are supplied with the LP software. They are shell procedures which interface between *lpsched* and devices. All models reside in the directory `/usr/spool/lp/model` and may be used as is with *lpadmin* `-m`. Models should have 644 permission if owned by `lp` and `bin`, or 664 permission if owned by `bin` and `bin`. Alternatively, LP administrators may modify copies of models and then use *lpadmin* `-i` to associate them with printers. The following list describes the *models* and lists the options which they may be given on the *lp* command line using the `-o` keyletter:

- dumb** interface for a line printer without special functions and protocol. Form feeds are assumed. This is a good model to copy and modify for printers which do not have models.
- 1640** DIABLO 1640 terminal running at 1200 baud, using XON/XOFF protocol. Options:
  - `-12` 12-pitch (10-pitch is the default)
  - `-f` do not use the 450(1) filter. The output has been pre-processed by either 450(1) or the *nroff*450 driving table.
- hp** Hewlett-Packard 2631A line printer at 2400 baud. Options:
  - `-c` compressed print
  - `-e` expanded print
- prx** Printronix P300 or P600 printer using XON/XOFF protocol at 1200 baud.

### EXAMPLE

1. Assuming there is an existing Hewlett-Packard 2631A line printer named *hp2*, it will use the **hp** model interface after the command:
 

```
/usr/lib/lpadmin -php2 -mhp
```
2. To obtain compressed print on *hp2*, use the command:
 

```
lp -dhp2 -o-c files
```
3. A DIABLO 1640 printer called *st1* can be added to the LP configuration with the command:
 

```
/usr/lib/lpadmin -pst1 -v/dev/tty20 -m1640
```
4. An *nroff* document may be printed on *st1* in any of the following ways:
 

```
nroff -T450 files | lp -dst1 -of
nroff -T450-12 files | lp -dst1 -of
nroff -T37 files | col | lp -dst1
```

5. The following command prints the password file on *stl* in 12-pitch:

```
lp -dstl -o12 /etc/passwd
```

*NOTE:* the **-12** option to the 1640 model should never be used in conjunction with *nroff*.

**FILES**

/usr/spool/lp/\*

**SEE ALSO**

accept(1M), enable(1), lp(1), lpsched(1M), lpstat(1), nroff(1).

**LPMOVE(1M)**

**SEE LPSCHED**

**LPMOVE(1M)**

**NAME**

*lpsched*, *lpshut*, *lpmove* — start/stop the LP request scheduler and move requests

**SYNOPSIS**

```
/usr/lib/lpsched  
/usr/lib/lpshut  
/usr/lib/lpmove requests dest  
/usr/lib/lpmove dest1 dest2
```

**DESCRIPTION**

*Lpsched* schedules requests taken by *lp(1)* for printing on line printers.

*Lpshut* shuts down the line printer scheduler. All printers that are printing at the time *lpshut* is invoked will stop printing. Requests that were printing at the time a printer was shut down will be reprinted in their entirety after *lpsched* is started again. All LP commands perform their functions even when *lpsched* is not running.

*Lpmove* moves requests that were queued by *lp(1)* between LP destinations. This command may be used only when *lpsched* is not running.

The first form of the command moves the named *requests* to the LP destination, *dest*. *Requests* are request ids as returned by *lp*. The second form moves all requests for destination *dest1* to destination *dest2*. As a side effect, *lp* will reject requests for *dest1*.

Note that *lpmove* never checks the acceptance status (see *accept(1M)*) for the new destination when moving requests.

**FILES**

```
/usr/spool/lp/*
```

**SEE ALSO**

*accept(1M)*, *enable(1)*, *lp(1)*, *lpadmin(1M)*, *lpstat(1)*.

**LPSHUT (1M)**

**SEE *LPSCHED***

**LPSHUT (1M)**

## NAME

mkfs - construct a file system

## SYNOPSIS

/etc/mkfs [ -s ] special blocks[:inodes] [gap blocks/cyl]

/etc/mkfs [ -s ] special proto [gap blocks/cyl]

## DESCRIPTION

*Mkfs* constructs a file system by writing on the special file according to the directions found in the remainder of the command line. The command waits 10 seconds before starting to construct the file system. If the second argument is given as a string of digits, *mkfs* builds a file system with a single empty directory on it. The size of the file system is the value of *blocks* interpreted as a decimal number. This is the number of *physical* disk blocks the file system will occupy. The boot program is left uninitialized. If the optional number of inodes is not given, the default is the number of *logical* blocks divided by 4.

If the *-s* option is specified, *mkfs* reports the maximum block number used (excluding free list blocks).

If the second argument is a file name that can be opened, *mkfs* assumes it to be a prototype file *proto*, and will take its directions from that file. The prototype file contains tokens separated by spaces or new-lines. The first token is the name of a file to be copied onto block zero as the bootstrap program. The second token is a number specifying the size of the created file system in *physical* disk blocks. Typically it will be the number of blocks on the device, perhaps diminished by space for swapping. The next token is the number of inodes in the file system. The maximum number of inodes configurable is 65500. The next set of tokens comprise the specification for the root file. File specifications consist of tokens giving the mode, the user ID, the group ID, and the initial contents of the file. The syntax of the contents field depends on the mode.

The mode token for a file is a 6-character string. The first character specifies the type of the file. (The characters *-bcd* specify regular, block special, character special and directory files respectively.) The second character of the type is either *u* or *-* to specify set-user-id mode or not. The third is *g* or *-* for the set-group-id mode. The rest of the mode is a three digit octal number giving the owner, group, and other read, write, execute permissions (see *chmod(1)*).

Two decimal number tokens come after the mode; they specify the user and group IDs of the owner of the file.

If the file is a regular file, the next token is a pathname whence the contents and size are copied. If the file is a block or character special file, two decimal number tokens follow which give the major and minor device numbers. If the file is a directory, *mkfs* makes the entries `.` and `..` and then reads a list of names and (recursively) files specifications for the entries in the directory. The scan is terminated with the token `$`.

A sample prototype specification follows:

```

/stand/diskboot
4872 110
d—777 3 1
usr      d—777 3 1
         sh      —755 3 1 /bin/sh
         ken     d—755 6 1
         $
         b0      b—644 3 1 0 0
         c0      c—644 3 1 0 0
         $
$

```

In both command syntaxes, the rotational *gap* and the number of *blocks/cyl* can be specified.

The *default* will be used if the supplied *gap* and *blocks/cyl* are considered illegal values or if a short argument count occurs.

#### EXAMPLE

```
mkfs /dev/fd0 2000 7 50
```

makes a file system in which 2000 is the total size of the file system to be put on */dev/fd0*; 7 is a sector interleave number which is used to stagger the disk blocks for more rapid reading, every 7 blocks, and 50 is a modulo operator that forces the sector interlace number first to allocate all blocks in the first 50 sectors, then the next 50, etc.

NOTE: The proper selection of the *m* and *n* parameters can improve disk efficiency. Disks which have full or partial track buffering should specify a *m* and *n* of 1 and 1; *m* and *n* for other disks must be determined by trial and error as the disk latency is related to rotational latency and cpu speed.

#### SEE ALSO

`chmod(1)`, `dir(4)`, `fs(4)`, `boot(8)`.

**BUGS**

If a prototype is used, it is not possible to initialize a file larger than 64K bytes, nor is there a way to specify links.

**NOTES**

The functionality of *mkfs1b* is now completely contained in *mkfs*, so *mkfs1b* no longer exists as a separate command.



**NAME**

`mklost+found` - make a lost+found directory for `fsck`

**SYNOPSIS**

`mklost+found`

**DESCRIPTION**

A directory `lost+found` is created in the current directory and a number of empty files are created therein and then removed so that there will be empty slots for `fsck(1M)`. This command should be run immediately after first mounting and changing directory to a newly created file system. For small file systems, it is sufficient (and much faster) to simply make a `lost+found` directory. Up to 30 files can be recovered in it.

**EXAMPLE**

`mklost+found`

in the current directory, creates a directory with empty slots named `lost+found`.

**SEE ALSO**

`fsck(1M)`, `mkfs(1M)`

**BUGS**

Should be done automatically by `mkfs`.

**NAME**

mknod — build special file

**SYNOPSIS**

```
/etc/mknod name c | b major minor  
/etc/mknod name p
```

**DESCRIPTION**

*Mknod* makes a directory entry and corresponding i-node for a special file. The first argument is the *name* of the entry. In the first case, the second is *b* if the special file is block-type (disks, tape) or *c* if it is character-type (other devices). The last two arguments are numbers specifying the *major* device type and the *minor* device (e.g. unit, drive, or line number), which may be either decimal or octal.

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

*Mknod* can also be used to create fifo's (a.k.a named pipes) (second case in *SYNOPSIS* above).

**EXAMPLE**

```
mknod /dev/tty4 c 3 4
```

would create file */dev/tty4* as a character special device with major number 3 and minor number 4.

**SEE ALSO**

mknod(2).

MONACCT(1M)

SEE ACCTSH

MONACCT(1M)

**NAME**

mount, umount — mount and dismount file system

**SYNOPSIS**

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

/etc/umount special

**DESCRIPTION**

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

These commands maintain a table of mounted devices. If invoked with no arguments, *mount* prints the table.

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

*Umount* announces to the system that the removable file system previously mounted on device *special* is to be removed.

**FILES**

/etc/mnttab mount table

**EXAMPLE**

```
mount /dev/xxxx /t
```

mounts device /dev/xxxx as file system /t.

**SEE ALSO**

setmnt(1M), mount(2), mnttab(4).

**DIAGNOSTICS**

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

*Umount* complains if the special file is not mounted or if it is busy. The file system is busy if it contains an open file or some user's working directory.

**BUGS**

Some degree of validation is done on the file system, however it is generally unwise to mount garbage file systems.

**NAME**

`mvdir` — move a directory

**SYNOPSIS**

`/etc/mvdir` *dirname* *name*

**DESCRIPTION**

*Mvdir* renames directories within a file system. *Dirname* must be a directory; *name* must not exist. Neither name may be a sub-set of the other (*/x/y* cannot be moved to */x/y/z*, nor vice versa).

Only super-user can use *mvdir*.

**EXAMPLE**

`mvdir dir1 dir2`

renames existing directory "dir1" to be a new directory "dir2".

**SEE ALSO**

`mkdir(1)`.

**NAME**

ncheck - generate names from i-numbers

**SYNOPSIS**

/etc/ncheck [ -i numbers ] [ -a ] [ -s ] [ file-system ]

**DESCRIPTION**

**N.B.:** For most normal file system maintenance, the function of *ncheck* is subsumed by *fsck(1M)*.

*Ncheck* with no argument generates a path name vs. i-number list of all files on a set of default file systems. Names of directory files are followed by */.* The *-i* option reduces the report to only those files whose i-numbers follow. The *-a* option allows printing of the names *.* and *..*, which are ordinarily suppressed. The *-s* option reduces the report to special files and files with set-user-ID mode; it is intended to discover concealed violations of security policy.

A file system may be specified.

The report is in no useful order, and probably should be sorted.

**EXAMPLE**

```
ncheck /dev/rdisk1
```

will report the pathnames and i-numbers of files on the specified device.

**SEE ALSO**

*fsck(1M)*, *sort(1)*.

**DIAGNOSTICS**

When the file system structure is improper, *??* denotes the "parent" of a parentless file and a path name beginning with *...* denotes a loop.

**NULLADM (1M)**

**SEE ACCTSH**

**NULLADM (1M)**

**POWERFAIL (1M)**

**SEE BRC**

**POWERFAIL (1M)**

**PRCTMP (1M)**

**SEE ACCTSH**

**PRCTMP (1M)**

**PRDAILY (1M)**

**SEE ACCTSH**

**PRDAILY (1M)**

**PRTACCT(1M)**

**SEE *ACCTSH***

**PRTACCT(1M)**



## NAME

pstat — print system facts

## SYNOPSIS

pstat [-abfimnprtuvx [suboptions]] [file]

## DESCRIPTION

*Pstat* interprets the contents of certain system tables. If *file* is given, the tables are sought there, otherwise in */dev/kmem*. Unless the *-n* option is used, the required namelist is taken from */unix*. Options are:

- a Under *-p*, describe all process slots rather than just active ones.
- b Print the system IO buffer header information with the following headings:
  - LOC The core location of the buffer header.
  - FLAGS
    - R The buffer is to be read.
    - W The buffer is to be written.
    - D The IO is done.
    - E An error occurred during the buffer's IO operation.
    - B The buffer is busy.
    - P The buffer is being used for physical (raw) IO.
    - M The buffer has map space allocated (not all machines).
    - W A process wants to access the buffer and is waiting for it.
    - A The buffer has aged.
    - Y The buffer is doing an asynchronous operation (the process that started the IO will not wait for it to complete).
    - L A buffer's contents have changed and it needs to be written out before it can be reallocated.
    - O The open routine has been called for this device.
    - S The buffer is "stale."
  - DEVICE The major and minor device numbers for the device to which the buffer is queued (or contains information from).
  - ADDR The core address of the data in the buffer.
  - BLKNO The block number of the block on DEVICE.
- f Print the open file table with these headings:
  - LOC The core location of this table entry.
  - FLG Miscellaneous state variables encoded thus:
    - R open for reading
    - W open for writing
    - P pipe
  - CNT Number of processes that know this open file.
  - INO The location of the inode table entry for this file.
  - OFFS The file offset, see *lseek(2)*.
- i Print the inode table with these headings:
  - LOC The core location of this table entry.
  - FLAGS Miscellaneous state variables encoded thus:
    - L locked
    - U update time *fs(4)* must be corrected
    - A access time must be corrected
    - M file system is mounted here

- W wanted by another process (L flag is on)  
 T contains a text file  
 C changed time must be corrected  
 CNT Number of open file table entries for this inode.  
 DEV Major and minor device number of file system in which this inode resides.  
 INO Inumber within the device.  
 MODE Mode bits, see `chmod(2)`.  
 NLK Number of links to this inode.  
 UID User ID of owner.  
 SIZ/DEV Number of bytes in an ordinary file, or major and minor device of special file.
- m Print information about core memory allocation and a dump of the memory free map with these headings:
- LOC The core address of the map entry.  
 ADDR The "click" address of the area this entry refers to.  
 SIZE The size of this area in "clicks".
- n Used to specify a namelist (system code file) other than the default of `/unix`. The next parameter specifies this file.
- p Print process table for active processes with these headings:
- LOC The core location of this table entry.  
 S Run state encoded thus:
- 0 no process  
 1 waiting for some event  
 3 runnable  
 4 being created  
 5 being terminated  
 6 stopped under trace
- F Miscellaneous state variables, or-ed together:
- 01 loaded  
 02 the scheduler process  
 04 locked  
 010 swapped out  
 020 traced  
 040 used in tracing  
 0100 locked in by `locking(2)`.
- PRI Scheduling priority, see `nice(2)`.  
 SIGNAL Signals received (signals 1-16 coded in bits 0-15),  
 UID Real user ID.  
 TIM Time resident in seconds; times over 127 coded as 127.  
 CPU Weighted integral of CPU time, for scheduler.  
 NI Nice level, see `nice(2)`.  
 PGRP Process number of root of process group (the opener of the controlling terminal).  
 PID The process ID number.  
 PPID The process ID of parent process.  
 ADDR If in core, the physical address of the "u-area" of the process measured in multiples of 64 bytes. If swapped out, the position in the swap area measured in multiples of 512 bytes.

- SIZE      Size of process image in multiples of 64 bytes.  
 WCHAN     Wait channel number of a waiting process.  
 LINK      Link pointer in list of runnable processes.  
 TEXTP     If text is pure, pointer to location of text table entry.  
 CLKT      Countdown for **alarm(2)** measured in seconds.
- r        Used to make the execution of **pstat** repeat at a rate defined by the next parameter.
- t        Print table for terminals (only DH11 and DL11 handled) with these headings:
- |          |   |
|----------|---|
| LOC      | Core location of this table entry.                            |
| RAW      | Number of characters in raw input queue.                      |
| CAN      | Number of characters in canonicalized input queue.            |
| OUT      | Number of characters in output queue.                         |
| PROC     | Core location of the proc routine.                            |
| IFLAG    | Input modes (see <i>termio(7)</i> ).                          |
| OFLAG    | Output modes (see <i>termio(7)</i> ).                         |
| CFLAG    | Control modes (see <i>termio(7)</i> ).                        |
| LFLAG    | Line discipline modes (see <i>termio</i> ).                   |
| STATE    | Internal state.   |
| TIMEOUT  | 00000001    Delay timeout in progress.                        |
| WOPEN    | 00000002    Waiting for open to complete.                     |
| ISOPEN   | 00000004    Device is open.                                   |
| TBLOCK   | 00000010  |
| CARR_ON  | 00000020    Software copy of carrier-present.                 |
| BUSY     | 00000040    Output in progress.                               |
| OASLP    | 00000100    Wakeup when output done.                          |
| IASLP    | 00000200    Wakeup when input done.                           |
| TTSTOP   | 00000400    Output stopped by CTRL-s.                         |
| EXTPROC  | 00001000    External processing.                              |
| TACT     | 00002000  |
| CLESC    | 00004000    Last char escape.                                 |
| RTO      | 00010000  |
| TTIOW    | 00020000  |
| TTXON    | 00040000  |
| TTXOFF   | 00100000  |
| TS_RCOLL | 00200000    Collision in read select.                         |
| TS_WCOLL | 00400000    Collision in write select.                        |
| TS_NBIO  | 01000000    Tty in non-blocking mode.                         |
| TS_ASYNC | 02000000    Tty in async I/O mode.                            |
| PGRP     | Process group for which this is controlling terminal.         |
| LN       | Line discipline.  |
| DEL      | Number of delimiters (newlines) in canonicalized input queue. |
| COL      | Calculated column position of terminal.                       |
| ROW      | Calculated row position of terminal.                          |
| IX       | Index to the table of core locations.                         |
- u        print information about a user process; the next argument is its address as given by **ps(1)**. The process must be in main memory, or the file used can be a core image and the address 0.

- v This option affects a number of the other options. It make them give a more verbose output. Often this means that they list table entries that are not currently active or in use.
- x Print the text table with these headings:
  - LOC The core location of this table entry.
  - FLAGS Miscellaneous state variables encoded thus:
    - T ptrace(2) in effect
    - W text not yet written on swap device
    - L loading in progress
    - K locked
    - w wanted (L flag is on)
  - DADDR Disk address in swap, measured in multiples of 512 bytes.
  - CADDR Core address, measured in multiples of core clicks (machine dependent).
  - SIZE Size of text segment, measured in multiples of core clicks (machine dependent).
  - IPTR Core location of corresponding inode.
  - CNT Number of processes using this text segment.
  - CCNT Number of processes in core using this text segment.

**FILES**

/unix namelist  
/dev/kmem default source of tables

**EXAMPLE**

`psstat -i`

displays all the active inodes in a table format with headings.

**SEE ALSO**

ps(1), stat(2), fs(4)

**UNIX Implementation** in the *Administrator Guide*.

## NAME

pstat — print system facts

## SYNOPSIS

pstat [ -abfimmprtuv [ suboptions ] ] [ file ]

## DESCRIPTION

*Pstat* interprets the contents of certain system tables. If *file* is given, the tables are sought there, otherwise in */dev/kmem*. Unless the *-n* option is used, the required namelist is taken from */unix*. Options are:

- a Under *-p*, describe all process slots rather than just active ones.
- b Print the system IO buffer header information with the following headings:
  - LOC The core location of the buffer header.
  - FLAGS
    - R The buffer is to be read.
    - W The buffer is to be written.
    - D The IO is done.
    - E An error occurred during the buffer's IO operation.
    - B The buffer is busy.
    - P The buffer is being used for physical (raw) IO.
    - M The buffer has map space allocated (not all machines).
    - W A process wants to access the buffer and is waiting for it.
    - A The buffer has aged.
    - Y The buffer is doing an asynchronous operation (the process that started the IO will not wait for it to complete).
    - L A buffer's contents have changed and it needs to be written out before it can be reallocated.
    - O The open routine has been called for this device.
    - S The buffer is "stale."
  - DEVICE The major and minor device numbers for the device to which the buffer is queued (or contains information from).
  - ADDR The core address of the data in the buffer.
  - BLKNO The block number of the block on DEVICE.
- f Print the open file table with these headings:
  - LOC The core location of this table entry.
  - FLG Miscellaneous state variables encoded thus:
    - R open for reading
    - W open for writing
    - P pipe
  - CNT Number of processes that know this open file.
  - INO The location of the inode table entry for this file.
  - OFFS The file offset, see *lseek(2)*.
- i Print the inode table with these headings:
  - LOC The core location of this table entry.
  - FLAGS Miscellaneous state variables encoded thus:
    - L locked
    - U update time *fs(4)* must be corrected
    - A access time must be corrected
    - M file system is mounted here
    - W wanted by another process (L flag is on)
    - T contains a text file
    - C changed time must be corrected
  - CNT Number of open file table entries for this inode.
  - DEVICE Major and minor device number of file system in which this inode resides.

- INO Inumber within the device.
- MODE Mode bits, see `chmod(2)`.
- NLK Number of links to this inode.
- UID User ID of owner.
- SIZE/DEV Number of bytes in an ordinary file, or major and minor device of special file.
- LOCK Address of the locklist structure for this inode.
- m Print information about core memory allocation and a dump of the memory free map with these headings:
- LOC The core address of the map entry.
- ADDR The "click" address of the area this entry refers to.
- SIZE The size of this area in "clicks".
- n Used to specify a namelist (system code file) other than the default of `/unix`. The next parameter specifies this file.
- p Print process table for active processes with these headings:
- LOC The core location of this table entry.
- S Run state encoded thus:
- 0 no process
  - 1 waiting for some event
  - 2 runnable
  - 3 being terminated
  - 4 stopped under trace
  - 5 being created
  - 6 running
  - 7 being xswapped
- F Flags (octal and additive) associated with the process:
- 0 swapped;
  - 1 system process;
  - 2 being traced by another process;
  - 4 another tracing flag;
  - 10 process cannot be woken by a signal;
  - 20 in core;
  - 40 locked in memory;
- PRI Scheduling priority, see `nice(2)`.
- SIGNAL Signals received (signals 1-16 coded in bits 0-15),
- UID Real user ID.
- TIM Time resident in seconds; times over 127 coded as 127.
- CPU Weighted integral of CPU time, for scheduler.
- NI Nice level, see `nice(2)`.
- PGRP Process number of root of process group (the opener of the controlling terminal).
- PID The process ID number.
- PPID The process ID of parent process.
- ADDR If in core, the physical address of the page tables in the proc structure for the "u-area" of the process. If swapped out, the position in the swap area measured in multiples of 512 bytes.
- SIZE Size of process image in multiples of logical page size.
- WCHAN Wait channel number of a waiting process.
- LINK Link pointer in list of runnable processes.
- CLKT Countdown for `alarm(2)` measured in seconds.

- r Used to make the execution of pstat repeat at a rate defined by the next parameter.
- t Print table for terminals (only DH11 and DL11 handled) with these headings:
 

LOC	Core location of this table entry.
RAW	Number of characters in raw input queue.
CAN	Number of characters in canonicalized input queue.
OUT	Number of characters in output queue.
PROC	Core location of the proc routine.
IFLAG	Input modes (see <i>termio(7)</i> ).
OFLAG	Output modes (see <i>termio(7)</i> ).
CFLAG	Control modes (see <i>termio(7)</i> ).
LFLAG	Line discipline modes (see <i>termio</i> ).
STATE	Internal state.
TIMEOUT	00000001 Delay timeout in progress.
WOPEN	00000002 Waiting for open to complete.
ISOPEN	00000004 Device is open.
TBLOCK	00000010
CARR_ON	00000020 Software copy of carrier-present.
BUSY	00000040 Output in progress.
OASLP	00000100 Wakeup when output done.
IASLP	00000200 Wakeup when input done.
TTSTOP	00000400 Output stopped by CTRL-s.
EXTPROC	00001000 External processing.
TACT	00002000
CLESC	00004000 Last char escape.
RTO	00010000
TTIOW	00020000
TTXON	00040000
TTXOFF	00100000
TS_RCOLL	00200000 Collision in read select.
TS_WCOLL	00400000 Collision in write select.
TS_NBIO	01000000 Tty in non-blocking mode.
TS_ASYNC	02000000 Tty in async I/O mode.
PGRP	Process group for which this is controlling terminal.
LN	Line discipline.
DEL	Number of delimiters (newlines) in canonicalized input queue.
COL	Calculated column position of terminal.
ROW	Calculated row position of terminal.
IX	Index to the table of core locations.
- u print information about a user process; the next argument is its address as given by ps(1). The process must be in main memory, or the file used can be a core image and the address 0.
- v This option affects a number of the other options. It make them give a more verbose output. Often this means that they list table entries that are not currently active or in use.

## FILES

/unix           namelist  
 /dev/kmem       default source of tables

**EXAMPLE****pstat -i**

displays all the active inodes in a table format with headings.

**SEE ALSO**

ps(1), stat(2), fs(4)

**UNIX Implementation** in the *Administrator Guide*.



**NAME**

*pwck*, *grpck* - password/group file checkers

**SYNOPSIS**

*/etc/pwck* [file]  
*/etc/grpck* [file]

**DESCRIPTION**

*Pwck* scans the password file and notes any inconsistencies. The checks include validation of the number of fields, login name, user ID, group ID, and whether the login directory and optional program name exist. The default password file is */etc/passwd*.

*Grpck* verifies all entries in the group file. This verification includes a check of the number of fields, group name, group ID, and whether all login names appear in the password file. The default group file is */etc/group*.

**EXAMPLE**

*pwck*  
will list inconsistencies in */etc/passwd*.  
*grpck*  
will list inconsistencies in */etc/group*.

**FILES**

*/etc/group*  
*/etc/passwd*

**SEE ALSO**

*group(4)*, *passwd(4)*.

**DIAGNOSTICS**

Group entries in */etc/group* with no login names are flagged.

RC(1M)

SEE BRC

RC(1M)

**NAME**

**reboot** – reboot the system

**SYNOPSIS**

*/etc/reboot*

**DESCRIPTION**

*Reboot* generates a Maintenance Reset Function (MRF), causing the processor to enter its system bootstrap code, thereby rebooting the system. It can be used to reboot the processor remotely, but this is practical only if a terminal line is enabled in */etc/inittab* for state 1 so that the file systems can be checked and state 2 entered.

*Reboot* will enter the boot sequence immediately, without flushing the internal system buffers. It must be used with extreme caution.

**REJECT (1M)**

**SEE ACCEPT**

**REJECT (1M)**

**NAME**

runacct — run daily accounting

**SYNOPSIS**

`/usr/lib/acct/runacct [mddd[state]]`

**DESCRIPTION**

*Runacct* is the main daily accounting shell procedure. It is normally initiated via *cron*(1M). *Runacct* processes *connect*, *fee*, *disk*, and process accounting files. It also prepares summary files for *prdaily* or billing purposes.

*Runacct* takes care not to damage active accounting files or summary files in the event of errors. It records its progress by writing descriptive diagnostic messages into *active*. When an error is detected, a message is written to */dev/console*, mail (see *mail*(1)) is sent to *root* and *adm*, and *runacct* terminates. *Runacct* uses a series of lock files to protect against re-invocation. The files *lock* and *lock1* are used to prevent simultaneous invocation, and *lastdate* is used to prevent more than one invocation per day.

*Runacct* breaks its processing into separate, restartable *states* using *statefile* to remember the last *state* completed. It accomplishes this by writing the *state* name into *statefile*. *Runacct* then looks in *statefile* to see what it has done and to determine what to process next. *States* are executed in the following order:

<b>SETUP</b>	Move active accounting files into working files.
<b>WTMPFIX</b>	Verify integrity of <i>wtmp</i> file, correcting date changes if necessary.
<b>CONNECT1</b>	Produce <i>connect</i> session records in <i>ctmp.h</i> format.
<b>CONNECT2</b>	Convert <i>ctmp.h</i> records into <i>taect.h</i> format.
<b>PROCESS</b>	Convert process accounting records into <i>taect.h</i> format.
<b>MERGE</b>	Merge the <i>connect</i> and process accounting records.
<b>FEES</b>	Convert output of <i>chargefee</i> into <i>taect.h</i> format and merge with <i>connect</i> and process accounting records.
<b>DISK</b>	Merge disk accounting records with <i>connect</i> , <i>process</i> , and <i>fee</i> accounting records.
<b>MERGETACCT</b>	Merge the daily total accounting records in <i>daytaect</i> with the summary total accounting records in <i>/usr/adm/acct/sum/taect</i> .
<b>CMS</b>	Produce command summaries.
<b>USEREXIT</b>	Any installation-dependent accounting programs can be included here.
<b>CLEANUP</b>	Cleanup temporary files and exit.

To restart *runacct* after a failure, first check the *active* file for diagnostics, then fix up any corrupted data files such as *pacct* or *wtmp*. The *lock* files and *lastdate* file must be removed before *runacct* can be restarted. The argument *mddd* is necessary if *runacct* is being restarted, and specifies the

month and day for which *runacct* will rerun the accounting. Entry point for processing is based on the contents of *statefile*; to override this, include the desired *state* on the command line to designate where processing should begin.

**EXAMPLE**

```
nohup runacct 2> /usr/adm/acct/nite/fd2log &
starts runacct.
nohup runacct 0601 2>> /usr/adm/acct/nite/fd2log &
restarts runacct.
nohup runacct 0601 MERGE 2>> /usr/adm/acct/nite/fd2log &
restarts runacct at a specific state.
```

**FILES**

```
/etc/wtmp
/usr/adm/pacct*
/usr/src/cmd/acct/tacct.h
/usr/src/cmd/acct/ctmp.h
/usr/adm/acct/nite/active
/usr/adm/acct/nite/daytaacct
/usr/adm/acct/nite/lock
/usr/adm/acct/nite/lock1
/usr/adm/acct/nite/lastdate
/usr/adm/acct/nite/statefile
/usr/adm/acct/nite/ptacct*.mmd
```

**SEE ALSO**

mail(1), acct(1M), acctems(1M), acctcom(1), acctcon(1M), acctmerg(1M), acctprc(1M), acctsh(1M), cron(1M), fwtmp(1M), acct(2), acct(4), utmp(4).

**ACCOUNTING** in the *Administrator Guide*.

**BUGS**

Normally it is not a good idea to restart *runacct* in the **SETUP** *state*. Run **SETUP** manually and restart via:

```
runacct mmd WTMPFIX
```

If *runacct* failed in the **PROCESS** *state*, remove the last *ptacct* file because it will not be complete.

**SA1 (1M)**

**SEE SAR**

**SA1 (1M)**

**SA2 (1M)**

**SEE SAR**

**SA2 (1M)**

**SADC (1M)**

**SEE SAR**

**SADC (1M)**

**NAME**

sa1, sa2, sadc — system activity report package

**SYNOPSIS**

```
/usr/lib/sa/sadc [t n] [ofile]
```

```
/usr/lib/sa/sa1 [t n]
```

```
/usr/lib/sa/sa2 [-ubdycwaqvmA] [-s time] [-e time] [-i sec]
```

**DESCRIPTION**

System activity data can be accessed at the special request of a user (see *sar(1)*) and automatically on a routine basis as described here. The operating system contains a number of counters that are incremented as various system actions occur. These include CPU utilization counters, buffer usage counters, disk and tape I/O activity counters, TTY device activity counters, switching and system-call counters, file-access counters, queue activity counters, and counters for inter-process communications.

*Sadc* and shell procedures, *sa1* and *sa2*, are used to sample, save, and process this data.

*Sadc*, the data collector, samples system data *n* times every *t* seconds and writes in binary format to *ofile* or to standard output. If *t* and *n* are omitted, a special record is written. This facility is used at system boot time to mark the time at which the counters restart from zero. The */etc/rc* entry:

```
su adm -c "/usr/lib/sa/sadc /usr/adm/sa/sa`date +%d`"
```

writes the special record to the daily data file to mark the system restart.

The shell script *sa1*, a variant of *sadc*, is used to collect and store data in binary file */usr/adm/sa/sadd* where *dd* is the current day. The arguments *t* and *n* cause records to be written *n* times at an interval of *t* seconds, or once if omitted. The entries in *crontab* (see *cron(1M)*):

```
0 * * 0,6 su adm -c "/usr/lib/sa/sa1"
0 8-17 * * 1-5 su adm -c "/usr/lib/sa/sa1 1200 3"
0 18-7 * * 1-5 su adm -c "/usr/lib/sa/sa1"
```

will produce records every 20 minutes during working hours and hourly otherwise.

The shell script *sa2*, a variant of *sar(1)*, writes a daily report in file */usr/adm/sa/sar<sub>dd</sub>*. The options are explained in *sar(1)*. The *crontab* entry:

```
5 18 * * 1-5 su adm -c "/usr/lib/sa/sa2 -s 8:00 -e 18:01 -i 3600 -A"
```

will report important activities hourly during the working day.



The structure of the binary daily data file is:

```

struct sa {
    struct sysinfo si;      /* see /usr/include/sys/sysinfo.h */
    int szinode;           /* current entries of i-node table */
    int szfile;            /* current entries of file table */
    int sztext;            /* current entries of text table */
    int szproc;            /* current entries of proc table */
    int mszinode;          /* size of i-node table */
    int mszfile;           /* size of file table */
    int msztext;           /* size of text table */
    int mszproc;           /* size of proc table */
    long inodeovf;         /* cumul. overflows of i-node table */
    long fileovf;          /* cumul. overflows of file table */
    long textovf;          /* cumul. overflows of text table */
    long procovf;          /* cumul. overflows of proc table */
    time_t ts;             /* time stamp, seconds */
    long devio[NDEVS][4]; /* device info for up to NDEVS units */
#define IO_OPS 0 /* cumul. I/O requests */
#define IO_BCNT 1 /* cumul. blocks transferred */
#define IO_ACT 2 /* cumul. drive busy time in ticks */
#define IO_RESP 3 /* cumul. I/O resp time in ticks */
};

```

#### FILES

```

/usr/adm/sa/sadd    daily data file
/usr/adm/sa/sar dd  daily report file
/tmp/sa.adrfl      address file

```

#### SEE ALSO

cron(1M), sag(1G), sar(1), timex(1).

## NAME

sa1, sa2, sadc - system activity report package

## SYNOPSIS

```
/usr/lib/sa/sadc [t n] [ofile]
```

```
/usr/lib/sa/sa1 [t n]
```

```
/usr/lib/sa/sa2 [-ubdycwaqvmA] [-stime] [-etime] [-lsec]
```

## DESCRIPTION

System activity data can be accessed at the special request of a user (see *sar(1)*) and automatically on a routine basis as described here. The operating system contains a number of counters that are incremented as various system actions occur. These include CPU utilization counters, buffer usage counters, disk and tape I/O activity counters, TTY device activity counters, switching and system-call counters, file-access counters, queue activity counters, and counters for inter-process communications.

*Sadc* and shell procedures, *sa1* and *sa2*, are used to sample, save, and process this data.

*Sadc*, the data collector, samples system data *n* times every *t* seconds and writes in binary format to *ofile* or to standard output. If *t* and *n* are omitted, a special record is written. This facility is used at system boot time to mark the time at which the counters restart from zero. The */etc/rc* entry:

```
su adm -c "/usr/lib/sa/sadc /usr/adm/sa/sa`date +%d`"
```

writes the special record to the daily data file to mark the system restart.

The shell script *sa1*, a variant of *sadc*, is used to collect and store data in binary file */usr/adm/sa/sadd* where *dd* is the current day. The arguments *t* and *n* cause records to be written *n* times at an interval of *t* seconds, or once if omitted. The entries in *crontab* (see *cron(1M)*):

```
0 * * * 0,6 su adm -c "/usr/lib/sa/sa1"
0 8-17 * * 1-5 su adm -c "/usr/lib/sa/sa1 1200 3"
0 18-7 * * 1-5 su adm -c "/usr/lib/sa/sa1"
```

will produce records every 20 minutes during working hours and hourly otherwise.

The shell script *sa2*, a variant of *sar(1)*, writes a daily report in file */usr/adm/sa/sardd*. The options are explained in *sar(1)*. The *crontab* entry:

```
5 18 * * 1-5 su adm -c "/usr/lib/sa/sa2 -s 8:00 -e 18:01 -i 3600 -A"
```

will report important activities hourly during the working day.

The structure of the binary daily data file is:

```

struct sa {
    struct sysinfo si;          /* defined in /usr/include/sys/sysinfo.h */
    struct minfo mi;          /* defined in /usr/include/sys/sysinfo.h */
    int szinode;              /* current size of inode table */
    int szfile;               /* current size of file table */
    int szproc;               /* current size of proc table */
    int szlckf;               /* current size of file record header table */
    int szlckr;               /* current size of file record lock table */
    int mszinode;             /* maximum size of inode table */
    int mszfile;              /* maximum size of file table */
    int mszproc;              /* maximum size of proc table */
    int mszlckf;              /* maximum size of file record header table */
    int mszlckr;              /* maximum size of file record lock table */
    long inodeovf;           /* cumulative overflows of inode table
                             since boot */
    long fileovf;            /* cumulative overflows of file table
                             since boot */
    long procovf;            /* cumulative overflows of proc table
                             since boot */
    time_t ts;               /* time stamp */

    int apstate;
    long devio[NDEVS][4];    /* device unit information */

#define IO_OPS                0 /* number of I/O requests since boot */
#define IO_BCNT              1 /* number of blocks transferred since boot */
#define IO_ACT                2 /* cumulative time in ticks when drive is active */
#define IO_RESP               3 /* cumulative I/O response time in ticks since boot */
};

```

#### FILES

```

/usr/adm/sa/sadd    daily data file
/usr/adm/sa/saradd daily report file
/tmp/sa.adrfl      address file

```

#### SEE ALSO

cron(1M), sag(1G), sar(1), timex(1).

**NAME**

setmnt - establish mount table

**SYNOPSIS**

/etc/setmnt

**DESCRIPTION**

*Setmnt* creates the **/etc/mnttab** table (see *mnttab(4)*), which is needed for both the *mount(1M)* and *umount* commands. *Setmnt* reads standard input and creates a *mnttab* entry for each line. Input lines have the format:

filesys node

where *filesys* is the name of the file system's *special file* (e.g., "dsk/?s?") and *node* is the root name of that file system. Thus *filesys* and *node* become the first two strings in the *mnttab(4)* entry.

**EXAMPLE**

```
/etc/devnm / |grep -vv swap |grep -v root | /etc/setmnt
```

will put an entry for the root file system and the device on which it is mounted into the file **/etc/mnttab** (except if it is mounted on a device named "swap" or "root").

**FILES**

/etc/mnttab

**SEE ALSO**

*mount(1M)*, *devnm(1M)*, *mnttab(4)*.

**BUGS**

Evil things will happen if *filesys* or *node* are longer than 32 characters. *Setmnt* silently enforces an upper limit on the maximum number of *mnttab* entries.

SHUTACCT(1M)

SEE ACCTSH

SHUTACCT(1M)

C

C

C

**NAME**

shutdown — terminate all processing

**SYNOPSIS**

/etc/shutdown

**DESCRIPTION**

*Shutdown* is part of the UNIX System operation procedures. Its primary function is to terminate all currently running processes in an orderly and cautious manner. The procedure is designed to interact with the operator (i.e., the person who invoked *shutdown*). *Shutdown* may instruct the operator to perform some specific tasks, or to supply certain responses before execution can resume. *Shutdown* goes through the following steps:

All users logged on the system are notified to log off the system by a broadcasted message. The operator may display his/her own message at this time. Otherwise, the standard file save message is displayed.

If the operator wishes to run the file-save procedure, *shutdown* unmounts all file systems.

All file systems' super blocks are updated before the system is brought to single-user mode. A sync must be done before re-booting the system, to ensure file system integrity.

**DIAGNOSTICS**

The most common error diagnostic that will occur is *device busy*. This diagnostic happens when a particular file system could not be unmounted.

**SEE ALSO**

mount(1M), sync(1).

**STARTUP(1M)**

**SEE ACCTSH**

**STARTUP(1M)**

**NAME**

swap - swap administrative interface

**SYNOPSIS**

*/etc/swap* -a *swapdev* *swaplow* *swaplen*

*/etc/swap* -d *swapdev* *swaplow*

*/etc/swap* -l

**DESCRIPTION**

*Swap* provides a method of adding, deleting, and monitoring the system swap areas used by the memory manager. The following options are recognized:

- a Add the specified swap area. *Swapdev* is the name of block special device, e.g., */dev/dsk/1s0*. *Swaplow* is the offset in 512-byte blocks into the device where the swap area should begin. *Swaplen* is the length of the swap area in 512-byte blocks. This option can only be used by the super-user. Swap areas are normally added by the system start up routine */etc/rc* when going into multi-user mode.
- d Delete the specified swap area. *Swapdev* is the name of block special device, e.g., */dev/dsk/1s0*. *Swaplow* is the offset in 512-byte blocks into the device where the swap area should begin. Using this option marks the swap area as "being deleted." The system will not allocate any new blocks from the area, and will try to free swap blocks from it. The area will remain in use until all blocks from it are freed. This option can only be used by the super-user.
- l List the status of all the swap areas. The output has four columns:
 

<b>DEV</b>	The <i>swapdev</i> special file for the swap area if one can be found in the <i>/dev/dsk</i> or <i>/dev</i> directories, and its major/minor device number in decimal.
<b>LOW</b>	The <i>swaplow</i> value for the area in 512-byte blocks.
<b>LEN</b>	The <i>swaplen</i> value for the area in 512-byte blocks.
<b>FREE</b>	The number of free 512-byte blocks in the area. If the swap area is being deleted, this column will be marked (indel).

**WARNINGS**

No check is done to see if a swap area being added overlaps with an existing swap area or file system.



**NAME**

*sysdef* — system definition

**SYNOPSIS**

*/etc/sysdef* [ *opsys* [ *master* ] ]

**DESCRIPTION**

*Sysdef* analyzes the named operating system file and extracts configuration information. This includes all hardware devices as well as system devices and all tunable parameters.

The output of *sysdef* can usually be used directly by *config(1M)* to regenerate the appropriate configuration files.

**FILES**

*/unix*            default operating system file  
*/etc/master*    default table for hardware specifications

**SEE ALSO**

*config(1M)*, *master(4)*.

**BUGS**

For devices that have interrupt vectors but are not interrupt-driven, the output of *sysdef* cannot be used for *config*. Because information regarding *config* aliases is not preserved by the system, device names returned might not be accurate.

**TAPESAVE (1M)**

**SEE *FILESAVE***

**TAPESAVE (1M)**

**TELINIT (1M)**

**SEE *INIT***

**TELINIT (1M)**

**NAME**

*tic* - terminfo compiler

**SYNOPSIS**

*tic* [ *-v* [*n*] ] file ...

**DESCRIPTION**

*Tic* translates terminfo files from the source format into the compiled format. The results are placed in the directory */usr/lib/terminfo*.

The *-v* (verbose) option causes *tic* to output trace information showing its progress. If the optional integer is appended, the level of verbosity can be increased.

*Tic* compiles all terminfo descriptions in the given files. When a *use=* field is discovered, *tic* searches first the current file, then the master file, which is *"/terminfo.src"*.

If the environment variable *TERMINFO* is set, the results are placed there instead of */usr/lib/terminfo*.

Some limitations: total compiled entries cannot exceed 4096 bytes. The name field cannot exceed 128 bytes.

**FILES**

*/usr/lib/terminfo/\*/\** compiled terminal capability data base

**SEE ALSO**

*curses(3X)*, *terminfo(4)*.

**BUGS**

Instead of searching *./terminfo.src*, it should check for an existing compiled entry.

**TURNACCT (1M)**

**SEE ACCTSH**

**TURNACCT (1M)**

**UMOUNT (1M)**

**SEE MOUNT**

**UMOUNT (1M)**

**UNLINK (1M)**

**SEE LINK**

**UNLINK (1M)**

**NAME**

updater — update files between two machines

**SYNOPSIS**

updater [ key ] local remote ...

**DESCRIPTION**

*Updater* updates files between two machines.

One of the following key letters must be included:

- t** Take files from the remote machine, updating the local machine.
- p** Put files from the local machine onto the remote machine, updating the remote machine.
- d** List the difference between files on the local and remote machines.

The following key letters are optional:

- u** Update a file only if it exists on both machines; this is the default condition.
- r** Replace a file if it did not exist on the destination machine.

*Local* refers to the local directory name.

*Remote* refers to the remote directory names. Only one remote name can be specified if the **p** (put) key is specified.

**ALGORITHM**

Open `/dev/tty0` to the remote machine.

*Stty* the local port and send a *stty* command to the remote machine to condition both ends of the connection.

Send a "cd remote ; sumdir . | sort +2 > /tmp/rXXXXXX" to remote machine for each remote system; "cd local ; sumdir . | sort > /tmp/lXXXXXX" for local machine.

Wait for remote to complete.

Take /tmp/rXXXXXX.

Do a comparison between the local and the union of the remotes:

exists on remote only:

If both the **t** and **r** keys are specified, take the file;  
otherwise list the file.

exists on local only:

If both **p** and **r** keys are specified, put the file;  
otherwise list the file.

exist on both but different:

If **t** key is specified, take the file.

If **p** key is specified, put the file.

If **d** key is specified, list the file.

same:

nothing

**EXAMPLE**

updater d . .

uses `/dev/tty0` to communicate with a remote machine, and compares directories on the remote and local systems.

**NAME**

uuclean - uucp spool directory clean-up

**SYNOPSIS**

`/usr/lib/uucp/uuclean [ options ]`

**DESCRIPTION**

*Uuclean* will scan the spool directory for files with the specified prefix and delete all those which are older than the specified number of hours.

The following options are available.

- d***directory* Clean *directory* instead of the spool directory. If *directory* is not a valid spool directory it cannot contain "work files" i.e., files whose names start with "C.". These files have special meaning to *uuclean* pertaining to *uucp* job statistics.
- p***pre* Scan for files with *pre* as the file prefix. Up to 10 **-p** arguments may be specified. A **-p** without any *pre* following will cause all files older than the specified time to be deleted.
- n***time* Files whose age is more than *time* hours will be deleted if the prefix test is satisfied. (default time is 72 hours)
- w***file* The default action for *uuclean* is to remove files which are older than a specified time (see **-n** option). The **-w** option is used to find those files older than *time* hours, however, the files are not deleted. If the argument *file* is present the warning is placed in *file*, otherwise, the warnings will go to the standard output.
- s***sys* Only files destined for system *sys* are examined. Up to 10 **-s** arguments may be specified.
- m***file* The **-m** option sends mail to the owner of the file when it is deleted. If a *file* is specified then an entry is placed in *file*.

This program is typically started by *cron*(1M).

**EXAMPLE**

```
uuclean -pT -pRC -n0 -m
```

removes all files in `/usr/spool/uucp` with a prefix of T or RC, and mails notifications to the owners of the removed files.

**FILES**

`/usr/lib/uucp` directory with commands used by *uuclean* internally  
`/usr/spool/uucp` spool directory

**SEE ALSO**

*cron*(1M), *uucp*(1C), *uux*(1C).

## NAME

uusub - monitor uucp network

## SYNOPSIS

/usr/lib/uucp/uusub [ options ]

## DESCRIPTION

*Uusub* defines a *uucp* subnetwork and monitors the connection and traffic among the members of the subnetwork. The following options are available:

- s* *sys* Add *sys* to the subnetwork.
- d* *sys* Delete *sys* from the subnetwork.
- l* Report the statistics on connections.
- r* Report the statistics on traffic amount.
- f* Flush the connection statistics.
- u* *hr* Gather the traffic statistics over the past *hr* hours.
- c* *sys* Exercise the connection to the system *sys*. If *sys* is specified as *all*, then exercise the connection to all the systems in the subnetwork.

The meanings of the connections report are:

*sys* #call #ok time #dev #login #nack #other

where *sys* is the remote system name, #*call* is the number of times the local system tries to call *sys* since the last flush was done, #*ok* is the number of successful connections, *time* is the latest successful connect time, #*dev* is the number of unsuccessful connections because of no available device (e.g. ACU), #*login* is the number of unsuccessful connections because of login failure, #*nack* is the number of unsuccessful connections because of no response (e.g. line busy, system down), and #*other* is the number of unsuccessful connections because of other reasons.

The meanings of the traffic statistics are:

*sfile* *sbyte* *rfile* *rbyte*

where *sfile* is the number of files sent and *sbyte* is the number of bytes sent over the period of time indicated in the latest *uusub* command with the *-u* *hr* option. Similarly, *rfile* and *rbyte* are the numbers of files and bytes received.

## EXAMPLE

uusub -c all -u 24

is typically started by *cron*(1M) once a day.

## FILES

/usr/spool/uucp/SYSLOG	system log file
/usr/lib/uucp/L_sub	connection statistics
/usr/lib/uucp/R_sub	traffic statistics

## SEE ALSO

uucp(1C), uustat(1C).

**NAME**

vchk — version checkup

**SYNOPSIS**

**vchk** [*argument*] ...

**DESCRIPTION**

*Vchk* is a highly specialized form of *make*(1) designed to check and maintain the modes, ownerships, and versions of a set of files specified in the *description file*. The description file is essentially a "photograph" of what a healthy system (i.e., one with all its components in the correct state) looks like. It contains a list of pathnames (for both files and directories) that should exist and have specific protections and contents. *Vchk* reads the description file, checks each item specified and prints error messages when a file does not match its description. Many problems can be fixed directly by *vchk*, such as incorrect mode and/or owner and missing link names. All other problems involve actually replacing a file, detected by comparing some combination of checksum, length, and/or version number (from the description file) with the value generated from the actual file being checked. When a file needs to be replaced *whk* invokes the command named by the *REMAKE* macro (see *MACROS* below).

Each *argument* is either a definition or an option. Option arguments begin with the character `-` and consist of a string of letters (called *flags*) from the set `'ADIPsabcdefiklmprstvX'`. The `f` and `t` flags cause the next argument to be considered specially. The `p` and `P` flags cause the rest of the argument in which they appear to be considered specially. Other arguments are either macro definitions (i.e., *name* = *string* pairs) or simply strings which are saved as numeric macros. Briefly, the *flags* are as follows:

- A** *sysid*  
specifies an alternate *sysid* rather than using the one found in `/etc/sys_id`.
- D** enables debugging messages.
- I** process control lines only.
- P** *sysid*  
preprocess the description file; *sysid* is optional and is explained below under *PREPROCESSING*.
- S** suppress printing of non-fatal error messages.
- a** checks all lines in the description file. Modifies the **b**, **P**, and **k** options.
- b** build a description file for the current directory.
- c** print shell commands to fix the file system.
- d** suppress re-installation commands and error messages.
- e** suppress checks for everything but existence.
- f** *filename*  
cause *whk* to read *filename* instead of `/etc/vchk_tree`.
- i** go interactive: modifies **b**, **c**, and **x** options.



- k perform checksums on files having checksum field.
- l suppress listing of files left in directories.
- m allow multiple copies of files.
- ppw *file*  
force *vchk* to re-evaluate and/or use an alternate password file.
- r allow redundant password entries (user ID1).
- s remain silent about trivial problems.
- v suppress checking of version numbers.
- x execute shell commands to fix the file system (cf. the *-c* option which prints rather than executes).

### DESCRIPTION FILE SYNTAX

Lines in the description file are either comments, control lines, specifications, or commands. Control lines provide a simple *ifdef* mechanism for selectively ignoring specification lines. Specification lines describe files and/or directories that need to be checked. Commands are not processed by *vchk* but (in the spirit of *make(1)*) are used when the file specified above them is found to need replacing.

Several conventions are observed to maintain the readability of the description file; for example, a trailing backslash and all leading white space on the following line are ignored when processed. In addition, backslash may be used to delay the expansion of macros (in macro definitions only) and, as described below, to alter the evaluation of parentheses and braces in pathnames.

### COMMENTS

Comment lines always start with a '#'. If the second character on a comment line is also a '#', then that line is printed on the standard error when read by *vchk*. Any line may have a trailing comment, which is universally ignored.

### CONTROL LINES

Control lines start with a '.' (period). The mechanism is similar to the C language pre-processor except that defined words do not have values associated with them; words are simply defined or not. The control commands supported are as follows:

#### *.define wordlist*

where *wordlist* is a list of white-space-separated words to be defined which have no associated values. Note that only the first twelve letters of defined words are significant. Storage for defined words is static. There is a maximum of sixty defined words at any given time.

#### *.ifdef define\_expression*

where *define\_expression* is an infix boolean expression involving defined words and the operators !, &, and |, which mean 'not', 'and', and 'or', respectively. The value of each word evaluates to a boolean "yes" if that word is defined and "no" if not. If the expression evaluates to be false, lines are ignored until the matching *.endif* or *.else* control line is read. *ifndef* is also supported and reverses the sense of the expression test.

*.include filename*

is very similar the C pre-processor *include* with the exception that there are no default searching places and that the filename is not enclosed in double quotes or angle brackets. In addition, if the first character of the filename is an exclamation mark (!), then the rest of line is considered to be a shell command and its standard output is what gets included.

*.undef wordlist*

undefines each word in the wordlist.

*.unset wordlist*

frees the storage associated with macro definitions (detailed in a following section) for the given wordlist (which is composed of macro names). Words in wordlist that are already unset (or were never set) are silently ignored.

*.chdir directory\_name*

changes the current directory to the one specified and alters the starting location of pathnames anchored from the current directory.

*.exit message*

causes *vchk* to print the message and exit immediately.

As an aid to debugging the description file, a single `!` on a line by itself causes *vchk* to print the currently defined control symbols on the standard error when it reads that line.

**MACROS**

A macro processing facility very similar to the one used by *make*(1) is provided. Macros are defined when a line containing the macro name, an equal sign, and the value is read. The value may be null or include macro invocations. Unlike `'defined'` words, macro names are fully significant and are saved in dynamic memory. Macros are invoked by the `'$'` character. As in *make* scripts, macro names must be surrounded by `()`s when they are longer than a single letter. There is a special macro (named `!`, thus referenced with `$.!`) which always expands to the name of the current directory. It is useful in the construction of link names since most files have their links close by.

Except in the definition of a new macro (where interpretation may need to be delayed) and in comments, it is always an error for a macro to be used if it is undefined. Since the `'##'` comment is printed after macro substitution, it is a useful debugging tool. In keeping with the spirit of the `'dump control words'` command (`!`), a single `'$'` on a line by itself prints all the currently defined macros and their values.

Note that lines are re-scanned once a macro has been substituted so that a macro may be defined to expand to a control line, comment, or even a macro definition. Note that this degenerates to a recursive loop if the definition of a macro contains a reference to that macro.

Two predefined macros exist. The first, called `REMAKE`, contains the name (and options) of the program to use to replace damaged or missing files. If the file `/etc/sys_id` exists and is not empty, it is assumed to contain the *UniSoft* code name assigned to your system and *vchk* will setup the `REMAKE` macro to be the command `"take -iN"` where `N` is the name

found in `/etc/sys_id`. This allows systems that reside at *UniSoft* to be updated automatically over a direct *tty* line via the *take(1C)* program. If the `/etc/sys_id` file has a single empty line in it, then **REMAKE** will be set to "take -i". This allows remote systems to be updated automatically over phone lines. If the `/etc/sys_id` file does not exist, *vchk* sets the **REMAKE** macro to be "install". Note that the description file may redefine the **REMAKE** macro at any time.

The second predefined macro is called **ARGS** and can be set but not referenced. Strings assigned to **ARGS** are treated as command line options. The - preceding option keyletters is still required, and enables that option. A plus sign, '+', must be used instead to disable a keyletter option. Resetting the **b**, **p**, and **r** is not allowed.

### DESCRIPTION LINES

Each line that is not a comment, control line, macro definition, or command is considered to be the specification for a particular file or directory. These have a simple and regular syntax: the first and only mandatory field is the *pathname*, which must begin at the root (`/`) or the current directory (`./`). In practice we find that starting all lines with a macro allows easy relocation of the entire tree described and is very readable.

The rest of the line contains optional information about the contents and protection of the file. Contents specifications are separated from the pathname by white space. The entire protection specification is bracketed to separate it from the rest of the line.

### PATHNAMES

Pathnames refer to directories (if and only if they end in a `/` character) or files (if they do not end in a `/`). Use of shell metacharacters (*globbing*) is not supported but two mechanisms are provided to allow variable pathnames: braces, `{}`s, and parentheses. Braces are interpreted just as in *cs(1)*; each of the expanded pathnames must exist and must match the description given. Parentheses in pathnames are interpreted similarly, except that *exactly one* of the resulting pathnames must exist. This feature is useful, for example, to allow a program to be in either `/bin` or `/usr/bin`, but not both.

Parentheses and braces are expanded left to right; for example, the construction `(a,b){x,y}` means either `ax` and `ay` must exist or `bx` and `by`. Backslashes may be used to delay or prevent the interpretation of `0`s and `{}`s. For example, `\(a,b){x,y}` means one of `ax` or `bx` and one of `ay` or `by` must exist. One layer of backslashes are removed for each pass through the pathname and each time an unescaped parenthesis or bracket is detected and expanded, another pass is made.

Note that when alternative paths are used (i.e., parentheses occur in the pathname) the first one is considered the one to be rebuilt in the event that all are missing. For example, the pathname `(/usr)/bin/lis` would look first in `/usr/bin` for `lis`, then in `/bin`, and try to "REMAKE `/usr/bin/lis`" if both are missing. The reverse is true for `(./usr)/bin/lis`.

### SPECIFICATIONS

Two kinds of specifications are implemented. The first kind deals with the contents of the file or directory and follows the pathname (separated from

it by white space). The second kind deals with the files protection; these are enclosed in some type of parentheses to separate them from contents specifications. The kind of parentheses used, ()s, []s, <>s, or {}s, modify the action taken by *vchk* according to the table below:

- () Enables checking and replacing of the file.
- [] Enables checking but never replacing. If the file is missing, *vchk* will complain but not try to rebuild it.
- <> Enables checking (if and only if the file exists or the *-a* command line option is given) and never replacing.
- { } Enables checking but not repairing, (i.e., if the file is missing then it will be remade, but if it exists and is incorrect it will not be remade).

Associated with each directory is a default mode and ownership for the files and directories contained within it. Unless explicitly given, each directory inherits its defaults from its parent directory. If unspecified, the uppermost directory (either the root or current directory) sets the mode and ownership of its contents from its own mode and ownership. These defaults may be reset at any time simply by following the directory name with a mode and/or user name.

Regular files have three optional contents specification fields: length, checksum, and version number. These may be specified with a word (either Length, Checksum, or Version), an optional space, or a numeric value. The word may be any prefix, for example, 'Length 34' or 'L34'. The checksums used are the same as those produced by *sum*(1). The length checked is that returned by *stat*(2). These checks do not apply to device files (only block and character devices are supported); thus their contents specification field must begin with either *b* or *c* and must be followed by the major and minor device number (separated by white space). If *x* is used instead of either the major or minor device number, *vchk* will allow the device to have any value.

The protection specification consists of a list of command prefixes separated by semicolons. The commands supported are *chmod*, *chown*, and *link to*. If angle brackets (i.e., <>s) are used instead of parentheses to enclose the protection specification, the file or directory so referenced is optional and will not generate diagnostics if it is missing. It can be raised to the status of []s by giving the *-a* command line option. If square brackets (i.e., []s) are used, the referenced file cannot be replaced automatically, as for example, the password file. If curly brackets (i.e., {}s) are used, the referenced file will be replaced only if it is missing, not if it exists and is wrong (according to the description file). This is useful for files like */etc/termcap*.

Any other information in the protection specification is treated as a special comment that is printed with error messages about that file.

## OPERATION

In order for *vchk* to check the ownership of files it must map user ID numbers onto login names. The password file is normally used for this mapping but it is too expensive to look up each name every time it is used so *vchk* creates a temporary file (*/etc/vchk\_pw*) the first time it is run; whenever its temporary is out of date with respect to the real password file,

*vchk* recreates the temporary file.

In the process of reading the password file *vchk* inspects each account and prints diagnostics when it finds questionable data there. These messages are warnings or simply situations which bear reporting; the format of these messages is "Line <number>: message". The word "Error:" is prepended to the warnings for a particular line if *vchk* has decided to ignore that line of the password file.

The *-p*{*pw\_file*} option is provided to allow users who do not have write permission in /etc to use *vchk*. If specified with a filename after it, *vchk* will get the saved password information from that file. If the file does not exist, *vchk* will create it. Use of the *-p* option without a filename informs *vchk* to reprocess the password file even if it is not out of date.

*Vchk* normally expects the description file to be /etc/*vchk\_tree*, but if the standard input to *vchk* is a regular file, that file will be read instead.

Instead of redirecting the standard input, the *-f* option can be used to respecify the description file. It is an error to use both.

The best way to build a new description file is to *chdir* to the appropriate directory and run *vchk* with the *-b* option. A description file for the current directory will be produced on the standard output. The *-i* option may also be used, causing *vchk* to ask before descending each directory.

In addition to reporting errors, the *-c* flag prints shell commands to correct the detected error. The *-i* option can be used with the *-c* option to ask before outputting a command.

It is inadvisable to use the *-x* flag until the description file has been used and debugged. This flag allows *vchk* to execute the *chmod*(1), *chown*(1), and *ln*(1) commands internally, saving much time. Re-installation commands (cf. the REMAKE macro) are executed via the *system*(3S) call.

## PREPROCESSING

The *-Psysid* command line option provides a means for simplifying a complex description file. Everything except macro substitution is suppressed and after each line is parsed, it is printed on the standard output instead of being used to check the filesystem.

If a *sysid* is given after the *-P* flag, then it is used to lookup a line from the (/etc/*takelist*) file. (See *take*(1C) for a more complete description of the function of the /etc/*takelist* file.)

The lines in /etc/*takelist* are composed of any number of fields (called alternates) separated by colons (:s). The first alternate in a line is a list of system names separated by or bars (|s). The *sysid* above is compared with each of the system names in the first alternate of each line until it is found. If not found, then *vchk* exits with an appropriate error message.

When the line from /etc/*takelist* for the current *sysid* is found, then each of the additional alternates are considered lists of root directories (separated by colons or bars) to be prepended to filenames in the tree file before looking for them.

If a file is found in more or less than exactly 1 place in the list, then an error is reported and that line is not include in the preprocessed output. If it

is found, then the checksum, length, and version number are computed from that file and replaced in the preprocessed output.

**EXAMPLE**

Following are some excerpts from a typical description file.

```

B = (/usr)/bin          # programs can be in /bin or /usr/bin
/                       bin 755          (chmod 755)
$B/ar                   Version 1.0
$B/awk                  Version 1.3
$B/more                 Version 1.0      (link to $./page)
$B/sccsdiff             C54686 L1253    (shell script)
$B/su                   Version 1.0      (chown root; chmod 4755)
/etc/                   root 644
/etc/passwd             [password file]
/etc/group              [group file]
/etc/init               Version 1.0      (chmod 700)
/etc/update             Version 1.1      (chmod 700)
/etc/ddate              <dump dates>

```

The first line of the above example defines a macro, *B*, to be the string *(/usr)/bin*. This macro is then invoked on lines 3 through 7 of the example to allow the programs mentioned to be in either */usr/bin* or */bin*.

The second line specifies that the root directory (*/*) should have mode that the default mode and owner for files found in it be *755* and *bin*.

The third line specifies that the *ar* program should be version 1.0, owned by *bin* and have mode *755*. The mode and owner are implied in the following way. Each directory inherits its mode and ownership from its parent. Thus */bin* inherits the owner of root (which is unspecified in the example and thus defaults to whatever the owner of the root (*/*) is when the example is run). The mode of the root directory is specified as *755*.

**FILES**

```

/etc/vchk_pw           the file where vchk saves the password file summary.
/etc/passwd           the password file.
/etc/vchk_tree         the default description file.
/dev/tty              where vchk prints questions and gets the responses
                      (when the -i option is used).
<standard error>     used to print all diagnostics.
<standard output>    used to print shell commands and the newly built
                      description file (when using the -b option).
<standard input>     considered the default description file if it is a regular
                      file.

```

**SEE ALSO**

*chmod(1)*, *ln(1)*, *chown(1)*.

**BUGS**

There is no way (except tediously via the *-i* option) to exclude directories from inspection when building a new description file. There is also no way to automatically update an existing description file (i.e., to tell *vchk* to fix

the description file instead of the filesystem).

## NAME

volcopy, labelit — copy file systems with label checking

## SYNOPSIS

```
/etc/volcopy [options] fsname special1 volname1 special2 volname2
/etc/labelit special [ fsname volume [ -n ] ]
```

## DESCRIPTION

*Volcopy* makes a literal copy of the file system using a blocksize matched to the device. *Options* are:

- a invoke a verification sequence requiring a positive operator response instead of the standard 10-second delay before the copy is made,
- s (default) invoke the DEL if wrong verification sequence.

Other *options* are used only with tapes:

- bpdensity bits-per-inch (i.e., 800/1600/6250),
- feetsize size of reel in feet (i.e., 1200/2400),
- reelnum beginning reel number for a restarted copy,
- buf use double buffered I/O.

The program requests length and density information if it is not given on the command line or is not recorded on an input tape label. If the file system is too large to fit on one reel, *volcopy* will prompt for additional reels. Labels of all reels are checked. Tapes may be mounted alternately on two or more drives. If *volcopy* is interrupted, it will ask if the user wants to quit or wants a shell. In the latter case, the user can perform other operations (e.g., *labelit*) and return to *volcopy* by exiting the new shell.

The *fsname* argument represents the mounted name (e.g., *root*, *u1*, etc.) of the filesystem being copied.

The *special* should be the physical disk section or tape (e.g., */dev/rdisk/1s5*, */dev/rmt/0m*, etc.).

The *volname* is the physical volume name (e.g., *pk3*, *t0122*, etc.) and should match the external label sticker. Such label names are limited to six or fewer characters. *Volname* may be — to use the existing volume name.

*Special1* and *volname1* are the device and volume from which the copy of the file system is being extracted. *Special2* and *volname2* are the target device and volume.

*Fsname* and *volname* are recorded in the last 12 characters of the superblock (char *fsname*[6], *volname*[6]);).

*Labelit* can be used to provide initial labels for unmounted disk or tape file systems. With the optional arguments omitted, *labelit* prints current label values. The *-n* option provides for initial labeling of new tapes only (this destroys previous contents).

## EXAMPLE

```
volcopy newsys /dev/rp15 1 /dev/rfd0 1
```

copies volume 1 of the file system labeled *newsys* mounted on */dev/rp15* onto volume 1 of */dev/rfd0*.

```
labelit /dev/rfd0 oldsys save
```

relabels the file system mounted on */dev/rfd0* with a new *fsname* of *oldsys* and a new *volname* of *save*.



**FILES**

`/etc/log/filesave.log` a record of file systems/volumes copied

**SEE ALSO**

`sh(1)`, `fs(4)`.

**BUGS**

Only device names beginning `/dev/rmt` are treated as tapes.

**NAME**

wall - write to all users

**SYNOPSIS**

/etc/wall

**DESCRIPTION**

*Wall* reads its standard input until an end-of-file. It then sends this message to all currently logged in users preceded by:

Broadcast Message from ...

It is used to warn all users, typically prior to shutting down the system.

The sender must be super-user to override any protections the users may have invoked (see *mesg(1)*).

**EXAMPLE**

wall

will broadcast the standard input to all users who are not protected against receiving messages by the *mesg* command.

**FILES**

/dev/tty\*

**SEE ALSO**

*mesg(1)*, *write(1)*.

**DIAGNOSTICS**

"Cannot send to ..." when the open on a user's tty file fails.

**NAME**

whodo - who is doing what

**SYNOPSIS**

/etc/whodo

**DESCRIPTION**

*Whodo* produces merged, reformatted, and dated output from the *who(1)* and *ps(1)* commands.

**EXAMPLE**

/etc/whodo

will return something like the following:

```
UNIX
co root 13:52
co 60 0:01 sh
co 61 0:01 ps
co 62 0:00 sh
```

**FILES**

etc/passwd

**SEE ALSO**

ps(1), who(1).

**WTMPFIX (1M)**

**SEE *FWTMP***

**WTMPFIX (1M)**

**NAME**

intro -- introduction to special files

**DESCRIPTION**

This section describes special files that refer to specific hardware peripherals and UNIX System device drivers. The names of the entries are generally derived from names for the hardware, as opposed to the names of the special files themselves. Characteristics of both the hardware device and the corresponding UNIX System device driver are discussed where applicable.

**BUGS**

While the names of the entries *generally* refer to vendor hardware names, in certain cases these names are seemingly arbitrary for various historical reasons.

**NAME**

error — error-logging interface

**DESCRIPTION**

Minor device 0 of the *error* driver is the interface between a process and the system's error-record collection routines. The driver may be opened only for reading by a single process with super-user permissions. Each read causes an entire error record to be retrieved; the record is truncated if the read request is for less than the record's length.

**FILES**

/dev/error special file

**SEE ALSO**

errdemon(1M).

**KMEM (7)**

**SEE MEM**

**KMEM (7)**

**NAME**

mem, kmem -- core memory

**DESCRIPTION**

*Mem* is a special file that is an image of the core memory of the computer. It may be used, for example, to examine, and even to patch the system.

Byte addresses in *mem* are interpreted as memory addresses. References to non-existent locations cause errors to be returned.

Examining and patching device registers is likely to lead to unexpected results when read-only or write-only bits are present.

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

**FILES**

/dev/mem  
/dev/kmem



**NULL (7)**

**NULL (7)**

**NAME**

null — the null file

**DESCRIPTION**

Data written on a null special file is discarded.

Reads from a null special file always return 0 bytes.

**FILES**

/dev/null

**NAME**

sxt — pseudo-device driver

**DESCRIPTION**

*Sxt* is a pseudo-device driver that interposes a discipline between the standard *tty* line disciplines and a real device driver. The standard disciplines manipulate *virtual tty* structures (channels) declared by the *sxt* driver. *Sxt* acts as a discipline manipulating a *real tty* structure declared by a real device driver. The *sxt* driver is currently only used by the *shl*(1) command.

Virtual *tty*s are named by inodes in the subdirectory */dev/sxt* and are allocated in groups of up to eight. To allocate a group, a program should exclusively open a file with a name of the form */dev/sxt/??0* (channel 0) and then execute a SXTIOCLINK *ioctl* call to initiate the multiplexing.

Only one channel, the *controlling* channel, can receive input from the keyboard at a time; others attempting to read will be blocked.

There are two groups of *ioctl*(2) commands supported by *sxt*. The first group contains the standard *ioctl* commands described in *termio*(7), with the addition of the following:

TIOCEXCL      Set *exclusive use* mode: no further opens are permitted until the file has been closed.

TIOCNXCL      Reset *exclusive use* mode: further opens are once again permitted.

The second group are directives to *sxt* itself. Some of these may only be executed on channel 0.

SXTIOCLINK    Allocate a channel group and multiplex the virtual *tty*s onto the real *ty*. The argument is the number of channels to allocate. This command may only be executed on channel 0. Possible errors include:

EINVAL      The argument is out of range.

ENOTTY      The command was not issued from a real *ty*.

ENXIO      *linesw* is not configured with *sxt*.

EBUSY      An SXTIOCLINK command has already been issued for this real *ty*.

ENOMEM      There is no system memory available for allocating the virtual *ty* structures.

EBADF      Channel 0 was not opened before this call.

SXTIOCSWTCH    Set the controlling channel. Possible errors include:

EINVAL      An invalid channel number was given.

EPERM      The command was not executed from channel 0.

SXTIOCWF      Cause a channel to wait until it is the controlling channel. This command will return the error, *EINVAL*, if an invalid channel number is given.

SXTIOCUBLK	Turn off the <code>loblk</code> control flag in the virtual tty of the indicated channel. The error <code>EINVAL</code> will be returned if an invalid number or channel 0 is given.
SXTIOCSTAT	Get the status (blocked on input or output) of each channel and store in the <code>sxtblock</code> structure referenced by the argument. The error <code>EFAULT</code> will be returned if the structure cannot be written.
SXTIOCTRACE	Enable tracing. This command has no effect if tracing is not configured.
SXTIOCNOTRACE	Disable tracing. This command has no effect if tracing is not configured.

**FILES**

<code>/dev/sxt/??{0-7}</code>	Virtual tty devices
<code>/usr/include/sys/sxt.h</code>	Driver specific definitions.

**SEE ALSO**

`shl(1)`, `stty(1)`, `ioctl(2)`, `open(2)`, `termio(7)`.

**NAME**

termio — general terminal interface

**DESCRIPTION**

This section describes both a particular special file and the general nature of the terminal interface.

The file `/dev/tty` is, in each process, a synonym for the control terminal associated with the process group of that process, if any. It is useful for programs or shell sequences that wish to be sure of writing messages on the terminal no matter how output has been redirected. It can also be used for programs that demand the name of a file for output, when typed output is desired and it is tiresome to find out what terminal is currently in use.

All of the asynchronous communications ports use the same general interface, no matter what hardware is involved. The remainder of this section discusses the common features of this interface.

When a terminal file is opened, it normally causes the process to wait until a connection is established. In practice, users' programs seldom open these files; they are opened by *getty* and become a user's standard input, output, and error files. The very first terminal file opened by the process group leader of a terminal file not already associated with a process group becomes the *control terminal* for that process group. The control terminal plays a special role in handling quit and interrupt signals, as discussed below. The control terminal is inherited by a child process during a *fork(2)*. A process can break this association by changing its process group using *setpgrp(2)*.

A terminal associated with one of these files ordinarily operates in full-duplex mode. Characters may be typed at any time, even while output is occurring, and are only lost when the system's character input buffers become completely full, which is rare, or when the user has accumulated the maximum allowed number of input characters that have not yet been read by some program. Currently, this limit is 256 characters. When the input limit is reached, all the saved characters are thrown away without notice.

Normally, terminal input is processed in units of lines. A line is delimited by a new-line (ASCII LF) character, an end-of-file (ASCII EOT) character, or an end-of-line character. This means that a program attempting to read will be suspended until an entire line has been typed. Also, no matter how many characters are requested in the read call, at most one line will be returned. It is not, however, necessary to read a whole line at once; any number of characters may be requested in a read, even one, without losing information.

During input, erase and kill processing is normally done. By default, the character `#` erases the last character typed, except that it will not erase beyond the beginning of the line. By default, the character `@` kills (deletes) the entire input line, and optionally outputs a new-line character. Both these characters operate on a key-stroke basis, independently of any backspacing or tabbing that may have been done. Both the erase and kill characters may be entered literally by preceding them with the escape character `(\)`. In this case the escape character is not read. The erase and kill

characters may be changed.

Certain characters have special functions on input. These functions and their default character values are summarized as follows:

- INTR** (Rubout or ASCII DEL) generates an *interrupt* signal which is sent to all processes with the associated control terminal. Normally, each such process is forced to terminate, but arrangements may be made either to ignore the signal or to receive a trap to an agreed-upon location; see *signal(2)*.
- SWTCH** (Control-z or ASCII SUB) is used by the job control facility, *shl*, to change the current layer to the control layer, (not on PDP-11).
- QUIT** (Control-| or ASCII FS) generates a *quit* signal. Its treatment is identical to the interrupt signal except that, unless a receiving process has made other arrangements, it will not only be terminated but a core image file (called *core*) will be created in the current working directory.
- ERASE** (#) erases the preceding character. It will not erase beyond the start of a line, as delimited by a NL, EOF, or EOL character.
- KILL** (@) deletes the entire line, as delimited by a NL, EOF, or EOL character.
- EOF** (Control-d or ASCII EOT) may be used to generate an end-of-file from a terminal. When received, all the characters waiting to be read are immediately passed to the program, without waiting for a new-line, and the EOF is discarded. Thus, if there are no characters waiting, which is to say the EOF occurred at the beginning of a line, zero characters will be passed back, which is the standard end-of-file indication.
- NL** (ASCII LF) is the normal line delimiter. It can not be changed or escaped.
- EOL** (ASCII NUL) is an additional line delimiter, like NL. It is not normally used.
- STOP** (Control-s or ASCII DC3) can be used to temporarily suspend output. It is useful with CRT terminals to prevent output from disappearing before it can be read. While output is suspended, STOP characters are ignored and not read.
- START** (Control-q or ASCII DC1) is used to resume output which has been suspended by a STOP character. While output is not suspended, START characters are ignored and not read. The start/stop characters can not be changed or escaped.

The character values for INTR, QUIT, SWTCH, ERASE, KILL, EOF, and EOL may be changed to suit individual tastes. The ERASE, KILL, and EOF characters may be escaped by a preceding \ character, in which case no special function is done.

When the carrier signal from the data-set drops, a *hang-up* signal is sent to all processes that have this terminal as the control terminal. Unless other arrangements have been made, this signal causes the processes to terminate. If the hang-up signal is ignored, any subsequent read returns with

an end-of-file indication. Thus, programs that read a terminal and test for end-of-file can terminate appropriately when hung up on.

When one or more characters are written, they are transmitted to the terminal as soon as previously-written characters have finished typing. Input characters are echoed by putting them in the output queue as they arrive. If a process produces characters more rapidly than they can be typed, it will be suspended when its output queue exceeds some limit. When the queue has drained down to some threshold, the program is resumed.

Several *ioctl(2)* system calls apply to terminal files. The primary calls use the following structure, defined in `<termio.h>`:

```
#define NCC      8
struct termio {
    unsigned short  c_iflag;    /* input modes */
    unsigned short  c_oflag;    /* output modes */
    unsigned short  c_cflag;    /* control modes */
    unsigned short  c_lflag;    /* local modes */
    char            c_line;      /* line discipline */
    unsigned char   c_cc[NCC];  /* control chars */
};
```

The special control characters are defined by the array `c_cc`. The relative positions and initial values for each function are as follows:

```
0  VINTR  DEL
1  VQUIT  FS
2  VERASE #
3  VKILL  @
4  VEOF   EOT
5  VEOL   NUL
6  reserved
7  SWTCH  NUL
```

The `c_iflag` field describes the basic terminal input control:

```
IGNBRK  0000001  Ignore break condition.
BRKINT  0000002  Signal interrupt on break.
IGNPAR  0000004  Ignore characters with parity errors.
PARMRK  0000010  Mark parity errors.
INPCK   0000020  Enable input parity check.
ISTRIP  0000040  Strip character.
INLCR   0000100  Map NL to CR on input.
IGNCR   0000200  Ignore CR.
ICRNL   0000400  Map CR to NL on input.
IUCLC   0001000  Map upper-case to lower-case on input.
IXON    0002000  Enable start/stop output control.
IXANY   0004000  Enable any character to restart output.
IXOFF   0010000  Enable start/stop input control.
```

If `IGNBRK` is set, the break condition (a character framing error with data all zeros) is ignored, that is, not put on the input queue and therefore not read by any process. Otherwise if `BRKINT` is set, the break condition will generate an interrupt signal and flush both the input and output queues. If `IGNPAR` is set, characters with other framing and parity errors are ignored.

If PARMRK is set, a character with a framing or parity error which is not ignored is read as the three-character sequence: 0377, 0, X, where X is the data of the character received in error. To avoid ambiguity in this case, if ISTRIP is not set, a valid character of 0377 is read as 0377, 0377. If PARMRK is not set, a framing or parity error which is not ignored is read as the character NUL (0).

If INPCK is set, input parity checking is enabled. If INPCK is not set, input parity checking is disabled. This allows output parity generation without input parity errors.

If ISTRIP is set, valid input characters are first stripped to 7-bits, otherwise all 8-bits are processed.

If INLCR is set, a received NL character is translated into a CR character. If IGNCR is set, a received CR character is ignored (not read). Otherwise if ICRNL is set, a received CR character is translated into a NL character.

If IUCLC is set, a received upper-case alphabetic character is translated into the corresponding lower-case character.

If IXON is set, start/stop output control is enabled. A received STOP character will suspend output and a received START character will restart output. All start/stop characters are ignored and not read. If IXANY is set, any input character, will restart output which has been suspended.

If IXOFF is set, the system will transmit START/STOP characters when the input queue is nearly empty/full.

The initial input control value is all-bits-clear.

The *c\_oflag* field specifies the system treatment of output:

OPOST	0000001	Postprocess output.
OLCUC	0000002	Map lower case to upper on output.
ONLCR	0000004	Map NL to CR-NL on output.
OCRNL	0000010	Map CR to NL on output.
ONOCR	0000020	No CR output at column 0.
ONLRET	0000040	NL performs CR function.
OFILL	0000100	Use fill characters for delay.
OFDEL	0000200	Fill is DEL, else NUL.
NLDLY	0000400	Select new-line delays:
NL0	0	
NL1	0000400	
CRDLY	0003000	Select carriage-return delays:
CR0	0	
CR1	0001000	
CR2	0002000	
CR3	0003000	
TABDLY	0014000	Select horizontal-tab delays:
TAB0	0	
TAB1	0004000	
TAB2	0010000	
TAB3	0014000	Expand tabs to spaces.
BSDLY	0020000	Select backspace delays:
BS0	0	
BS1	0020000	

```

VTDLY    0040000  Select vertical-tab delays:
VT0      0
VT1      0040000
FFDLY    0100000  Select form-feed delays:
FF0      0
FF1      0100000

```

If OPOST is set, output characters are post-processed as indicated by the remaining flags, otherwise characters are transmitted without change.

If OLCUC is set, a lower-case alphabetic character is transmitted as the corresponding upper-case character. This function is often used in conjunction with IUCLC.

If ONLCR is set, the NL character is transmitted as the CR-NL character pair. If OCRNL is set, the CR character is transmitted as the NL character. If ONOCR is set, no CR character is transmitted when at column 0 (first position). If ONLRET is set, the NL character is assumed to do the carriage-return function; the column pointer will be set to 0 and the delays specified for CR will be used. Otherwise the NL character is assumed to do just the line-feed function; the column pointer will remain unchanged. The column pointer is also set to 0 if the CR character is actually transmitted.

The delay bits specify how long transmission stops to allow for mechanical or other movement when certain characters are sent to the terminal. In all cases a value of 0 indicates no delay. If OFILL is set, fill characters will be transmitted for delay instead of a timed delay. This is useful for high baud rate terminals which need only a minimal delay. If OFDEL is set, the fill character is DEL, otherwise NUL.

If a form-feed or vertical-tab delay is specified, it lasts for about 2 seconds.

New-line delay lasts about 0.10 seconds. If ONLRET is set, the carriage-return delays are used instead of the new-line delays. If OFILL is set, two fill characters will be transmitted.

Carriage-return delay type 1 is dependent on the current column position, type 2 is about 0.10 seconds, and type 3 is about 0.15 seconds. If OFILL is set, delay type 1 transmits two fill characters, and type 2, four fill characters.

Horizontal-tab delay type 1 is dependent on the current column position. Type 2 is about 0.10 seconds. Type 3 specifies that tabs are to be expanded into spaces. If OFILL is set, two fill characters will be transmitted for any delay.

Backspace delay lasts about 0.05 seconds. If OFILL is set, one fill character will be transmitted.

The actual delays depend on line speed and system load.

The initial output control value is all bits clear.

The `c_cflag` field describes the hardware control of the terminal:

```

CBAUD    0000017  Baud rate:
B0       0        Hang up
B50      0000001  50 baud
B75      0000002  75 baud

```



B110	0000003	110 baud
B134	0000004	134.5 baud
B150	0000005	150 baud
B200	0000006	200 baud
B300	0000007	300 baud
B600	0000010	600 baud
B1200	0000011	1200 baud
B1800	0000012	1800 baud
B2400	0000013	2400 baud
B4800	0000014	4800 baud
B9600	0000015	9600 baud
EXTA	0000016	External A
EXTB	0000017	External B
CSIZE	0000060	Character size:
CS5	0	5 bits
CS6	0000020	6 bits
CS7	0000040	7 bits
CS8	0000060	8 bits
CSTOPB	0000100	Send two stop bits, else one.
CREAD	0000200	Enable receiver.
PARENB	0000400	Parity enable.
PARODD	0001000	Odd parity, else even.
HUPCL	0002000	Hang up on last close.
CLOCAL	0004000	Local line, else dial-up.
LOBLK	0010000	Block layer output.

The CBAUD bits specify the baud rate. The zero baud rate, B0, is used to hang up the connection. If B0 is specified, the data-terminal-ready signal will not be asserted. Normally, this will disconnect the line. For any particular hardware, impossible speed changes are ignored.

The CSIZE bits specify the character size in bits for both transmission and reception. This size does not include the parity bit, if any. If CSTOPB is set, two stop bits are used, otherwise one stop bit. For example, at 110 baud, two stops bits are required.

If PARENB is set, parity generation and detection is enabled and a parity bit is added to each character. If parity is enabled, the PARODD flag specifies odd parity if set, otherwise even parity is used.

If CREAD is set, the receiver is enabled. Otherwise no characters will be received.

If HUPCL is set, the line will be disconnected when the last process with the line open closes it or terminates. That is, the data-terminal-ready signal will not be asserted.

If CLOCAL is set, the line is assumed to be a local, direct connection with no modem control. Otherwise modem control is assumed.

If LOBLK is set, the output of a job control layer will be blocked when it is not the current layer. Otherwise the output generated by that layer will be multiplexed onto the current layer.

The initial hardware control value after open is B300, CS8, CREAD, HUPCL.

The *c\_flag* field of the argument structure is used by the line discipline to control terminal functions. The basic line discipline (0) provides the following:

ISIG	0000001	Enable signals.
ICANON	0000002	Canonical input (erase and kill processing).
XCASE	0000004	Canonical upper/lower presentation.
ECHO	0000010	Enable echo.
ECHOE	0000020	Echo erase character as BS-SP-BS.
ECHOK	0000040	Echo NL after kill character.
ECHONL	0000100	Echo NL.
NOFLSH	0000200	Disable flush after interrupt or quit.

If ISIG is set, each input character is checked against the special control characters INTR, SWTCH, and QUIT. If an input character matches one of these control characters, the function associated with that character is performed. If ISIG is not set, no checking is done. Thus these special input functions are possible only if ISIG is set. These functions may be disabled individually by changing the value of the control character to an unlikely or impossible value (e.g., 0377).

If ICANON is set, canonical processing is enabled. This enables the erase and kill edit functions, and the assembly of input characters into lines delimited by NL, EOF, and EOL. If ICANON is not set, read requests are satisfied directly from the input queue. A read will not be satisfied until at least MIN characters have been received or the timeout value TIME has expired between characters. This allows fast bursts of input to be read efficiently while still allowing single character input. The MIN and TIME values are stored in the position for the EOF and EOL characters, respectively. The time value represents tenths of seconds.

If XCASE is set, and if ICANON is set, an upper-case letter is accepted on input by preceding it with a \ character, and is output preceded by a \ character. In this mode, the following escape sequences are generated on output and accepted on input:

<i>for:</i>	<i>use:</i>
\	\/
	\/
{	\/{
}	\/}
\	\/\

For example, A is input as \a, \n as \\n, and \N as \\\n.

If ECHO is set, characters are echoed as received.

When ICANON is set, the following echo functions are possible. If ECHO and ECHOE are set, the erase character is echoed as ASCII BS SP BS, which will clear the last character from a CRT screen. If ECHOE is set and ECHO is not set, the erase character is echoed as ASCII SP BS. If ECHOK is set, the NL character will be echoed after the kill character to emphasize that the line will be deleted. Note that an escape character preceding the erase or kill character removes any special function. If ECHONL is set, the NL character will be echoed even if ECHO is not set. This is useful for

terminals set to local echo (so-called half duplex). Unless escaped, the EOF character is not echoed. Because EOT is the default EOF character, this prevents terminals that respond to EOT from hanging up.

If NOFLSH is set, the normal flush of the input and output queues associated with the quit, switch, and interrupt characters will not be done.

The initial line-discipline control value is all bits clear.

The primary *ioctl(2)* system calls have the form:

```
ioctl (fildes, command, arg)
struct termio *arg;
```

The commands using this form are:

TCGETA	Get the parameters associated with the terminal and store in the <i>termio</i> structure referenced by <i>arg</i> .
TCSETA	Set the parameters associated with the terminal from the structure referenced by <i>arg</i> . The change is immediate.
TCSETAW	Wait for the output to drain before setting the new parameters. This form should be used when changing parameters that will affect output.
TCSETAF	Wait for the output to drain, then flush the input queue and set the new parameters.

Additional *ioctl(2)* calls have the form:

```
ioctl (fildes, command, arg)
int arg;
```

The commands using this form are:

TCSBRK	Wait for the output to drain. If <i>arg</i> is 0, then send a break (zero bits for 0.25 seconds).
TCXONC	Start/stop control. If <i>arg</i> is 0, suspend output; if 1, restart suspended output.
TCFLSH	If <i>arg</i> is 0, flush the input queue; if 1, flush the output queue; if 2, flush both the input and output queues.

One last *ioctl(2)* call has the form

```
ioctl (fildes, command, pArg)
int *pArg;
```

The command using this form is:

FIONREAD	Return the number of characters currently in a terminal's input buffer into the integer pointer <i>*pArg</i> .
----------	--

## FILES

```
/dev/tty
/dev/tty*
/dev/console
```

## SEE ALSO

```
stty(1), fork(2), ioctl(2), setpgrp(2), signal(2).
```

**NAME**

tty — controlling terminal interface

**DESCRIPTION**

The file `/dev/tty` is, in each process, a synonym for the control terminal associated with the process group of that process, if any. It is useful for programs or shell sequences that wish to be sure of writing messages on the terminal no matter how output has been redirected. It can also be used for programs that demand the name of a file for output, when typed output is desired and it is tiresome to find out what terminal is currently in use.

**FILES**

`/dev/tty`  
`/dev/tty*`

**NAME**

intro — introduction to system maintenance procedures

**DESCRIPTION**

This section outlines procedures that will be of interest to those charged with the task of system maintenance. Included are discussions on the topics of boot procedures and recovery from crashes.

**BUGS**

No manual can take the place of good, solid experience.

**NAME**

boot — startup procedures

**DESCRIPTION**

A 68000 UNIX system is typically started by a two-stage process. The first is a primary bootstrap which is used to read in the system itself.

The primary bootstrap, when read into memory and executed, sets up memory management if necessary, and types a prompt message on the console. Then it reads from the console a device specification (see below) followed immediately by a pathname. This program finds the corresponding file on the given device, loads that file into the proper memory location, and then transfers control of the program. Normal line editing characters can be used.

Conventionally, the name of the current version of the system is `'/unix'`. Then, the recipe is:

- 1) Load the boot program by fiddling with the console keys and crt as appropriate for your hardware.
- 2) When the `“?”` prompt is given, type [for example]
 

```

fd(0,0)unix
or
hd(0,0)unix

```
- 3 depending on whether you are loading from floppy or hard disk, respectively. The first 0 indicates the physical unit number; the second indicates the block number of the beginning of the logical file system (`device`) to be searched. (See below).
- 3 When asked for the device name, a list of valid device names can be obtained by typing a `“?”` followed by a carriage return. A carriage return by itself boots the UNIX system on the default device.

When the system is running, it types a `“#”` prompt. After doing any file system checks via `fsck(1M)` and setting the date (`date(1)`), the system can be brought up for standard operation by typing `init 2` in response to the `“#”` prompt, then an EOT (`control-d`) when the system requests it.

**Device Specifications**

A device specification has the following form:

```
3 device(unit,offset)
```

where *device* is the type of the device to be searched, *unit* is the unit number of the device, and *offset* is the block offset of the file system on the device. Device specifications vary according to which 68000 UNIX system you are using. Check manufacturer's instructions for the device specifications.

For example, the specification

```
3 hp(1,7000)
```

would indicate an HP disk, unit 1, and the file system found starting at block 7000.

**ROM Programs**

Programs to call the primary bootstrap may be installed in read-only

memories or manually keyed into main memory. Each program is position-independent but should be placed well above location 0 so it will not be overwritten. See manufacturer's instructions for a manually keyed-in ROM boot program, should one become necessary.

**FILES**

/unix — system code

## NAME

crash — what to do when the system crashes

## DESCRIPTION

This entry gives at least a few clues about how to proceed if the system crashes. It can't pretend to be complete.

In restarting after a crash, always bring up the system single-user, as specified in *boot(8)* as modified for your particular installation. Then perform an *fsck(1M)* on all file systems which could have been in use at the time of the crash. If any serious file system problems are found, they should be repaired. When you are satisfied with the health of your disks, check and set the date if necessary, then come up multi-user.

To even boot UNIX at all, certain files (and the directories leading to them) must be intact. First, the initialization program */etc/init* must be present and executable. For *init* to work correctly, */dev/console*, */bin/sh* and */bin/env* must be present. If one of these does not exist, the symptom is best described as thrashing. *Init* will go into a *forklexec* loop trying to create a Shell with proper standard input and output. The file */etc/rc* should also be there and be executable; the system will come up but will not be fully initialized without it.

If you cannot get the system to boot, a runnable system must be obtained from a backup medium. The root file system may then be doctored as a mounted file system as described below. If there are any problems with the root file system, it is probably prudent to go to a backup system to avoid working on a mounted file system.

*Repairing disks.* The first rule to keep in mind is that an addled disk should be treated gently; it shouldn't be mounted unless necessary, and if it is very valuable yet in quite bad shape, perhaps it should be copied before trying surgery on it. This is an area where experience and informed courage count for much.

*Fsck(1M)* is adept at diagnosing and repairing file system problems. It first identifies all of the files that contain bad (out of range) blocks or blocks that appear in more than one file. Any such files are then identified by name and *fsck* requests permission to remove them from the file system. Files with bad blocks should be removed. In the case of duplicate blocks, all of the files except the most recently modified should be removed. The contents of the survivor should be checked after the file system is repaired to ensure that it contains the proper data. (Note that running *fsck* with the *-n* option will cause it to report all problems without attempting any repair.)

*Fsck* will also report on incorrect link counts and will request permission to adjust any that are erroneous. In addition, it will reconnect any files or directories that are allocated but have no file system references to a "lost+found" directory. Finally, if the free list is bad (out of range, missing, or duplicate blocks) *fsck* will, with the operators concurrence, construct a new one.

*Why did it crash?* UNIX types a message on the console typewriter when it voluntarily crashes. Here is the current list of such messages, with enough information to provide a hope at least of the remedy. The message has the



form "panic: ...", possibly accompanied by other information. Left unstated in all cases is the possibility that hardware or software error produced the message in some unexpected way. Not all systems produce all of these panics.

**bflush: bad free list.**

A buffer management error occurred during a sync or umount system call

**blkdev**

The *getblk* routine was called with a nonexistent major device as argument. Definitely hardware or software error.

**devtab**

Null device table entry for the major device used as argument to *getblk*. Definitely hardware or software error.

**dpfrelse**

The list of processes currently mapped into the memory management unit has been lost (68451 only).

**iinit**

An I/O error reading the super-block for the root file system during initialization.

**interrupt stack overflow**

The kernel ran out of stack space on an interrupt. Subroutine depth is too great or too many local variables.

**kernel memory management error**

Bus error or address error in supervisor mode. Can be a software or hardware problem.

**kernel parity error**

A memory parity error occurred while the cpu was in supervisor mode.

**lost vmap segment**

An error occurred opening a shared memory segment.

**no data pages**

A memory management error occurred allocating mmu registers for a processes data segment.

**no fs**

A device has disappeared from the mounted-device table. Definitely hardware or software error.

**no imt**

Like "no fs", but produced elsewhere.

**no clock**

During initialization, neither the line nor programmable clock was found to exist.

**no procs**

Process table has been destroyed.

**no text page**

A memory management error occurred allocating mmu registers for a processes text segment.

**I/O error in swap**

An unrecoverable I/O error during a swap. Really shouldn't be a panic, but it is hard to fix.

**oops!!! syscall**

The interrupt vector for system calls is missing.

**out of swap space**

A program needs to be swapped out, and there is no more swap space. It has to be increased. This really shouldn't be a panic, but there is no easy fix.

**timeout table overflow**

The timeout table overflowed. The timeout table is not large enough or some routine is starting up too many timeouts.

**trap**

An unexpected trap has occurred within the system. This is accompanied by the following information:

**trap type**

2	bus error
3	address error
4	illegal instruction
5	divide by zero
6	CHK instruction
7	TRAPV instruction
8	privilege violation
9	trace
10	1010 emulator
11	1111 emulator
12-255	unexpected interrupt

virtual address (for bus/address errors only)

physical address

instruction register

function code

mmu dump

program counter

status register

program id

registers

**unexpected kernel trap**

A buserr or similar unexpected exception occurred while the cpu was in supervisor mode.

In some of these cases it is possible for hex 1000 to be added into the trap type; this indicates that the processor was in user mode when the trap occurred.

**SEE ALSO**

fsck(1M), boot(8).

## NAME

`delivermail` – deliver mail to arbitrary people

## SYNOPSIS

```
/etc/delivermail [ -[fr] address ] [ -a ] [ -e[empqw] ] [ -n ] [ -m ] [ -s ] [ -i ] [ -h N ] address ...
```

## DESCRIPTION

*Delivermail* delivers a letter to one or more people, routing the letter over whatever networks are necessary. *Delivermail* will do inter-net forwarding as necessary to deliver the mail to the correct place.

*Delivermail* is not intended as a user interface routine; it is expected that other programs will provide user-friendly front ends, and *delivermail* will be used only to deliver pre-formatted messages.

*Delivermail* reads its standard input up to a control-D or a single dot and sends a copy of the letter found there to all of the addresses listed. If the `-i` flag is given, single dots are ignored. It determines the network to use based on the syntax of the addresses. Addresses containing the character “@” or the word “at” are sent to BNET; and addresses containing “!” are sent to the UUCP net. Other addresses are assumed to be local.

Local addresses are looked up in the file `/usr/lib/aliases` and aliased appropriately. Aliasing can be prevented by preceding the address with a backslash or using the `-n` flag. Normally the sender is not included in any alias expansions, e.g., if “john” sends to “group”, and “group” includes “john” in the expansion, then the letter will not be delivered to “john”. The `-m` flag disables this suppression.

*Delivermail* computes the person sending the mail by looking at your login name. The “from” person can be explicitly specified by using the `-f` flag; or, if the `-a` flag is given, *delivermail* looks in the body of the message for a “From:” or “Sender:” field in ARPANET format. The `-f` and `-a` flags can be used only by the special users *root* and *network*, or if the person you are trying to become is the same as the person you are. The `-r` flag is entirely equivalent to the `-f` flag; it is provided for ease of interface only.

The `-ex` flag controls the disposition of error output, as follows:

- e Print errors on the standard output, and echo a copy of the message when done. It is assumed that a network server will return the message back to the user.

- m Mail errors back to the user.
- p Print errors on the standard output.
- q Throw errors away; only exit status is returned.
- w Write errors back to the user's terminal, but only if the user is still logged in and write permission is enabled; otherwise errors are mailed back.

If the error is not mailed back, and if the mail originated on the machine where the error occurred, the letter is appended to the file "dead.letter" in the sender's home directory.

If the first character of the user name is a vertical bar, the rest of the user name is used as the name of a program to pipe the mail to. It may be necessary to quote the name of the user to keep *delivermail* from suppressing the blanks from between arguments.

The message is normally edited to eliminate "From" lines that might confuse other mailers. In particular, "From" lines in the header are deleted, and "From" lines in the body are prepended by ">". The -s flag saves "From" lines in the header.

The -h flag gives a "hop-count", i.e., a measure of how many times this message has been processed by *delivermail* (presumably on different machines). Each time *delivermail* processes a message, it increases the hop-count by one; if it exceeds 30 *delivermail* assumes that an alias loop has occurred and it aborts the message. The hop-count defaults to zero.

*Delivermail* returns an exit status describing what it did. The codes are defined in *mailxits.h*:

- |                  |  |
|------------------|--|
| 0 EX_OK          | Successful completion on all addresses.                  |
| 2 EX_NOUSER      | User name not recognized.                                |
| 3 EX_UNAVAILABLE | Catchall meaning necessary resources were not available. |
| 4 EX_SYNTAX      | Syntax error in address.                                 |
| 5 EX_SOFTWARE    | Internal software error, including bad arguments.        |
| 6 EX_OSERR       | Temporary operating system error, such as "cannot fork". |
| 7 EX_NOHOST      | Host name not recognized.                                |

## FILES

*/usr/lib/aliases* to alias names  
*/bin/mail* to deliver local mail  
*/etc/netmailer* to deliver BNET mail  
*/bin/mail* to deliver UUCP mail (*/bin/mail* knows how...)  
*/tmp/mail\** temp file  
*/tmp/xscript\** saved transcript  
*/dev/log* to log status (optional)

## SEE ALSO

mail(1), aliases(4), netmailer(8N).

## BUGS

*Delivermail* sends one copy of the letter to each user; it should send one copy of the letter to each host and distribute to multiple users there whenever possible.

*Delivermail* assumes the addresses can be represented as one word. This is incorrect according to the ARPANET mail protocol RFC 733 (NIC 41952), but is consistent with the real world.

**NAME**

ftpd – DARPA Internet File Transfer Protocol server

**SYNOPSIS**

/etc/ftpd [-d] [-l] [-timeout]

**DESCRIPTION**

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

If the `-d` option is specified, each socket created will have debugging turned on (SO\_DEBUG). With debugging enabled, the system will trace all TCP packets sent and received on a socket. The program *trpt*(8N) may then be used to interpret the packet traces.

If the `-l` option is specified, each ftp session is logged on the standard output. This allows a line of the form `/etc/ftpd -l > /tmp/ftpllog` to be used to conveniently maintain a log of ftp sessions.

The ftp server will timeout an inactive session after 60 seconds. If the `-t` option is specified, the inactivity timeout period will be set to *timeout*.

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

<b>Request</b>	<b>Description</b>
ACCT	specify account (ignored)
ALLO	allocate storage (vacuously)
APPE	append to a file
CWD	change working directory
DELE	delete a file
HELP	give help information
LIST	give list files in a directory (ls -lg)
MODE	specify data transfer <i>mode</i>
NLST	give name list of files in directory (ls)
NOOP	do nothing
PASS	specify password
PORT	specify data connection port
QUIT	terminate session
RETR	retrieve a file
STOR	store a file
STRU	specify data transfer <i>structure</i>

TYPE	specify data transfer <i>type</i>
USER	specify user name
XCUP	change to parent of current working directory
XCWD	change working directory
XPWD	print the current working directory

The remaining ftp requests specified in Internet RFC 765 are recognized, but not implemented.

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

*Ftpd* authenticates users according to three rules.

- 1) The user name must be in the password data base, */etc/passwd*, and not have a null password. In this case a password must be provided by the client before any file operations may be performed.
- 2) The user name must not appear in the file */etc/ftpusers*.
- 3) If the user name is "anonymous" or "ftp", an anonymous ftp account must be present in the password file (user "ftp"). In this case the user is allowed to log in by specifying any password (by convention this is given as the client host's name).

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

`~ftp)`

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

`~ftp/bin)`

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

`~ftp/etc)`

Make this directory owned by the super-user and unwritable by anyone. The files *passwd*(4) and *group*(4) must be present for the *ls* command to work properly. These files should be mode 444.

`~ftp/pub)`

Make this directory mode 777 and owned by "ftp". Users should then

place files which are to be accessible via the anonymous account in this directory.

**SEE ALSO**

ftp(1N).

**BUGS**

There is no support for aborting commands.

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

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



**NAME**

*ifconfig* — configure network interface parameters

**SYOPSIS**

*/etc/ifconfig* interface [ *address* ] [ *parameters* ]

**DESCRIPTION**

*Ifconfig* is used to assign an address to a network interface and/or configure network interface parameters. *Ifconfig* must be used at boot time to define the network address of each interface present on a machine; it may also be used at a later time to redefine an interface's address. The *interface* parameter is a string of the form "name unit", e.g., "en0", while the address is either a host name present in the host name data base, *hosts*(4N), or a DARPA Internet address expressed in the Internet standard "dot notation".

The following parameters may be set with *ifconfig*:

- up** Mark an interface "up".
- down** Mark an interface "down". When an interface is marked "down", the system will not attempt to transmit messages through that interface.
- trailers** Enable the use of a "trailer" link level encapsulation when sending (default). If a network interface supports *trailers*, the system will, when possible, encapsulate outgoing messages in a manner which minimizes the number of memory to memory copy operations performed by the receiver.
- trailers** Disable the use of a "trailer" link level encapsulation.
- arp** Enable the use of the Address Resolution Protocol in mapping between network level addresses and link level addresses (default). This is currently implemented for mapping between DARPA Internet addresses and 10Mb/s Ethernet addresses.
- arp** Disable the use of the Address Resolution Protocol.

*Ifconfig* displays the current configuration for a network interface when no optional parameters are supplied.

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

**DIAGNOSTICS**

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

**SEE ALSO**

*intro*(5N), *netstat*(1N)

**NAME**

netmail — the B-NET network mail system

**DESCRIPTION**

The B-NET network mail system consists of the following programs:

**/etc/bnetmaild**

a simple mail daemon run by crontab or by hand. Looks in the *mail spool directory (/usr/spool/netmail)* for files to send out onto the network. Uses *remsh* to send mail to remote hosts. Deletes mail if the mail is apparently successfully sent. Deletes mail found lying around which is more than one week old; apparently the destination host in this case is off the net. Accepts no arguments.

**/etc/delivermail**

*exec'd* by */bin/mail* to deliver mail to users or networks depending on the contents of the address of the mail. If the address contains an

Ⓜ then deliver to the B-NET network, else if the address has a

! then deliver to the uucp network,

else deliver locally.

**/etc/netmailer**

*exec'd* by */etc/delivermail* to "deliver" netmail. Mail is actually deposited in */usr/spool/netmail* with appropriate network mail headers prepended. */etc/bnetmaild* actually sends the mail to the network. See *netmailer(8N)* for a description of flag arguments.

**/bin/mail**

has been modified to *exec /etc/delivermail (delivermail(8N))* which does aliasing and re-routing of mail destined for the b-network.

**FILES**

*/usr/spool/netmail*  
directory for network mail

*/usr/spool/netmail/bnetXXXXXX*  
actual mail file(s), XXXXXX = pid.

**SEE ALSO**

*mail(1)*, *remsh(1N)*, *delivermail(8N)*, *netmailer(8N)*.

**BUGS**

Many, no doubt; for example, lots of work should be done on */etc/bnetmaild*, i.e., if a piece of mail is deleted due to its being old or otherwise undeliverable, notification should be sent to the originator. Soon, however, *bnetmaild* will be replaced by *sendmail*.

**NAME**

netmailer - deliver mail to B-NET

**SYNOPSIS**

/etc/netmailer from-address to-host to-user

**DESCRIPTION**

*Netmailer* queues the letter found on its standard input for delivery to the host and user specified. The actual delivery will be performed by the B-NET mailer daemon (/etc/bnetmaild).

If the letter does not appear to have a full B-NET header, *netmailer* will insert "Date:" and "From:" fields in the proper format. The "From:" person is determined by the *from-address* argument, with colons translated to periods and "@<local-host>" appended. The "<local-host>" is obtained from the file /usr/lib/uucp/SYSTEMNAME.

**FILES**

/usr/spool/netmail/\*

**SEE ALSO**

delivermail(8N), netmail(8N).

**NAME**

remshd - remote shell server

**SYNOPSIS**

*/etc/remshd*

**DESCRIPTION**

*Remshd* is the server for the *rcmd*(3N) routine and, consequently, for the *remsh*(1N) program. The server provides remote execution facilities with authentication based on privileged port numbers.

*Remshd* listens for service requests at the port indicated in the "cmd" service specification; see *services*(4N). When a service request is received the following protocol is initiated:

- 1) The server checks the client's source port. If the port is not in the range 0-1023, the server aborts the connection.
- 2) The server reads characters from the socket up to a null ('\0') byte. The resultant string is interpreted as an ASCII number, base 10.
- 3) If the number received in step 1 is non-zero, it is interpreted as the port number of a secondary stream to be used for the *stderr*. A second connection is then created to the specified port on the client's machine. The source port of this second connection is also in the range 0-1023.
- 4) The server checks the client's source address. If the address is associated with a host for which no corresponding entry exists in the host name data base (see *hosts*(4N)), the server aborts the connection.
- 5) A null terminated user name of at most 16 characters is retrieved on the initial socket. This user name is interpreted as a user identity to use on the server's machine.
- 6) A null terminated user name of at most 16 characters is retrieved on the initial socket. This user name is interpreted as the user identity on the client's machine.
- 7) A null terminated command to be passed to a shell is retrieved on the initial socket. The length of the command is limited by the upper bound on the size of the system's argument list.
- 8) *Remshd* then validates the user according to the following steps. The remote user name is looked up in the password file and a *chdir* is performed to the user's home directory. If either the lookup or *chdir* fail, the connection is terminated. If the user is not the super-user, (user id 0), the file */etc/hosts.equiv* is consulted for a list of hosts considered "equivalent". If the client's host name is present in this file, the authentication is considered successful. If the lookup fails, or the user is the super-user, then the file *.rhosts* in the home directory of the remote user is checked for the machine name and identity of the user on the client's machine. If this lookup fails, the connection is terminated.
- 9) A null byte is returned on the connection associated with the *stderr* and the command line is passed to the normal login shell of the user. The shell inherits the network connections established by *remshd*.

**DIAGNOSTICS**

All diagnostic messages are returned on the connection associated with the `stderr`, after which any network connections are closed. An error is indicated by a leading byte with a value of 1 (0 is returned in step 9 above upon successful completion of all the steps prior to the command execution).

**"locuser too long"**

The name of the user on the client's machine is longer than 16 characters.

**"remuser too long"**

The name of the user on the remote machine is longer than 16 characters.

**"command too long"**

The command line passed exceeds the size of the argument list (as configured into the system).

**"Hostname for your address unknown."**

No entry in the host name database existed for the client's machine.

**"Login incorrect."**

No password file entry for the user name existed.

**"No remote directory."**

The `chdir` command to the home directory failed.

**"Permission denied."**

The authentication procedure described above failed.

**"Can't make pipe."**

The pipe needed for the `stderr`, wasn't created.

**"Try again."**

A `fork` by the server failed.

**"/bin/sh: ..."**

The user's login shell could not be started.

**SEE ALSO**

`remsh(1N)`, `rcmd(3N)`.

**BUGS**

The authentication procedure used here assumes the integrity of each client machine and the connecting medium. This is insecure, but is useful in an "open" environment.

A facility to allow all data exchanges to be encrypted should be present.

**NAME**

rexecd — remote execution server

**SYNOPSIS**

/etc/rexecd

**DESCRIPTION**

*Rexecd* is the server for the *rexec(3N)* routine. The server provides remote execution facilities with authentication based on user names and encrypted passwords.

*Rexecd* listens for service requests at the port indicated in the "exec" service specification; see *services(4N)*. When a service request is received the following protocol is initiated:

- 1) The server reads characters from the socket up to a null ('\0') byte. The resultant string is interpreted as an ASCII number, base 10.
- 2) If the number received in step 1 is non-zero, it is interpreted as the port number of a secondary stream to be used for the *stderr*. A second connection is then created to the specified port on the client's machine.
- 3) A null terminated user name of at most 16 characters is retrieved on the initial socket.
- 4) A null terminated, encrypted, password of at most 16 characters is retrieved on the initial socket.
- 5) A null terminated command to be passed to a shell is retrieved on the initial socket. The length of the command is limited by the upper bound on the size of the system's argument list.
- 6) *Rexecd* then validates the user as is done at login time and, if the authentication was successful, changes to the user's home directory, and establishes the user and group protections of the user. If any of these steps fail the connection is aborted with a diagnostic message returned.
- 7) A null byte is returned on the connection associated with the *stderr* and the command line is passed to the normal login shell of the user. The shell inherits the network connections established by *rexecd*.

**DIAGNOSTICS**

All diagnostic messages are returned on the connection associated with the *stderr*, after which any network connections are closed. An error is indicated by a leading byte with a value of 1 (0 is returned in step 7 above upon successful completion of all the steps prior to the command execution).

"username too long"

The name is longer than 16 characters.

"password too long"

The password is longer than 16 characters.

"command too long"

The command line passed exceeds the size of the argument list (as configured into the system).

**"Login incorrect."**

No password file entry for the user name existed.

**"Password incorrect."**

The wrong password was supplied.

**"No remote directory."**

The *chdir* command to the home directory failed.

**"Try again."**

A *fork* by the server failed.

**"/bin/sh: ..."**

The user's login shell could not be started.

**BUGS**

Indicating "Login incorrect" as opposed to "Password incorrect" is a security breach which allows people to probe a system for users with null passwords.

A facility to allow all data exchanges to be encrypted should be present.

**NAME**

rlogind -- remote login server

**SYNOPSIS**

/etc/rlogind [-d]

**DESCRIPTION**

*Rlogind* is the server for the *rlogin*(1N) program. The server provides a remote login facility with authentication based on privileged port numbers.

*Rlogind* listens for service requests at the port indicated in the "login" service specification; see *services*(4N). When a service request is received the following protocol is initiated:

- 1) The server checks the client's source port. If the port is not in the range 0-1023, the server aborts the connection.
- 2) The server checks the client's source address. If the address is associated with a host for which no corresponding entry exists in the host name data base (see *hosts*(4N)), the server aborts the connection.

Once the source port and address have been checked, *rlogind* allocates a pseudo terminal (see *pty*(5)), and manipulates file descriptors so that the slave half of the pseudo terminal becomes the *stdin*, *stdout*, and *stderr* for a login process. The login process is an instance of the *login*(1) program, invoked with the *-r* option. The login process then proceeds with the authentication process as described in *remsha*(8N), but if automatic authentication fails, it reprompts the user to login as one finds on a standard terminal line.

The parent of the login process manipulates the master side of the pseudo terminal, operating as an intermediary between the login process and the client instance of the *rlogin* program. In normal operation, the packet protocol described in *pty*(5) is invoked to provide *^S/^Q* type facilities and propagate interrupt signals to the remote programs. The login process propagates the client terminal's baud rate and terminal type, as found in the environment variable, "TERM"; see *environ*(5).

**DIAGNOSTICS**

All diagnostic messages are returned on the connection associated with the *stderr*, after which any network connections are closed. An error is indicated by a leading byte with a value of 1.

"Hostname for your address unknown."

No entry in the host name database existed for the client's machine.

"Try again."

A *fork* by the server failed.

"/bin/sh: ..."

The user's login shell could not be started.

**BUGS**

The authentication procedure used here assumes the integrity of each client machine and the connecting medium. This is insecure, but is useful in an "open" environment.



**NAME**

*route* — manually manipulate the routing tables

**SYNOPSIS**

*/etc/route* [ *-f* ] [ *command args* ]

**DESCRIPTION**

*Route* is a program used to manually manipulate the network routing tables. It normally is not needed, as the system routing table management daemon, *routed*(8N), should tend to this task.

*Route* accepts three commands: *add*, to add a route; *delete*, to delete a route; and *change*, to modify an existing route.

All commands have the following syntax:

*/etc/route command destination gateway [ metric ]*

where *destination* is a host or network for which the route is "to", *gateway* is the gateway to which packets should be addressed, and *metric* is an optional count indicating the number of hops to the *destination*. If no *metric* is specified, *route* assumes a value of 0. Routes to a particular host are distinguished from those to a network by interpreting the Internet address associated with *destination*. If the *destination* has a "local address part" of INADDR\_ANY, then the route is assumed to be to a network; otherwise, it is presumed to be a route to a host. If the route is to a destination connected via a gateway, the *metric* should be greater than 0. All symbolic names specified for a *destination* or *gateway* are looked up first in the host name database, *hosts*(4N). If this lookup fails, the name is then looked for in the network name database, *networks*(4N).

*Route* uses a raw socket and the SIOCADDRT and SIOCDELRT *ioctl*'s to do its work. As such, only the super-user may modify the routing tables.

If the *-f* option is specified, *route* will "flush" the routing tables of all gateway entries. If this is used in conjunction with one of the commands described above, the tables are flushed prior to the command's application.

**DIAGNOSTICS**

"add %s: gateway %s flags %x"

The specified route is being added to the tables. The values printed are from the routing table entry supplied in the *ioctl* call.

"delete %s: gateway %s flags %x"

As above, but when deleting an entry.

"%s %s done"

When the *-f* flag is specified, each routing table entry deleted is indicated with a message of this form.

"not in table"

A delete operation was attempted for an entry which wasn't present in the tables.

"routing table overflow"

An add operation was attempted, but the system was low on resources and was unable to allocate memory to create the new entry.

**SEE ALSO**

*intro*(5N), *routed*(8N).

**ROUTE (8N)**

**UniSoft**

**ROUTE (8N)**

**BUGS**

The change operation is not implemented, one should add the new route, then delete the old one.

**NAME**

routed - network routing daemon

**SYNOPSIS**

/etc/routed [-s] [-q] [-t] [ *logfile* ]

**DESCRIPTION**

*Routed* is invoked at boot time to manage the network routing tables. The routing daemon uses a variant of the Xerox NS Routing Information Protocol in maintaining up to date kernel routing table entries.

In normal operation *routed* listens on *udp*(SP) socket 520 (decimal) for routing information packets. If the host is an internetwork router, it periodically supplies copies of its routing tables to any directly connected hosts and networks.

When *routed* is started, it uses the *SIOCGIFCONF ioctl* to find those directly connected interfaces configured into the system and marked "up" (the software loopback interface is ignored). If multiple interfaces are present, it is assumed the host will forward packets between networks. *Routed* then transmits a *request* packet on each interface (using a broadcast packet if the interface supports it) and enters a loop, listening for *request* and *response* packets from other hosts.

When a *request* packet is received, *routed* formulates a reply based on the information maintained in its internal tables. The *response* packet generated contains a list of known routes, each marked with a "hop count" metric (a count of 16, or greater, is considered "infinite"). The metric associated with each route returned provides a metric *relative to the sender*.

*Response* packets received by *routed* are used to update the routing tables if one of the following conditions is satisfied:

- (1) No routing table entry exists for the destination network or host, and the metric indicates the destination is "reachable" (i.e. the hop count is not infinite).
- (2) The source host of the packet is the same as the router in the existing routing table entry. That is, updated information is being received from the very internetwork router through which packets for the destination are being routed.
- (3) The existing entry in the routing table has not been updated for some time (defined to be 90 seconds) and the route is at least as cost effective as the current route.
- (4) The new route describes a shorter route to the destination than the one currently stored in the routing tables; the metric of the new route is compared against the one stored in the table to decide this.

When an update is applied, *routed* records the change in its internal tables and generates a *response* packet to all directly connected hosts and networks. *Routed* waits a short period of time (no more than 30 seconds) before modifying the kernel's routing tables to allow possible unstable situations to settle.

In addition to processing incoming packets, *routed* also periodically checks the routing table entries. If an entry has not been updated for 3 minutes, the entry's metric is set to infinity and marked for deletion. Deletions are delayed an additional 60 seconds to insure the invalidation is propagated

throughout the internet.

Hosts acting as internetwork routers gratuitously supply their routing tables every 30 seconds to all directly connected hosts and networks.

Supplying the *-s* option forces *routed* to supply routing information whether it is acting as an internetwork router or not. The *-q* option is the opposite of the *-s* option. If the *-t* option is specified, all packets sent or received are printed on the standard output. In addition, *routed* will not divorce itself from the controlling terminal so that interrupts from the keyboard will kill the process. Any other argument supplied is interpreted as the name of file in which *routed's* actions should be logged. This log contains information about any changes to the routing tables and a history of recent messages sent and received which are related to the changed route.

In addition to the facilities described above, *routed* supports the notion of "distant" *passive* and *active* gateways. When *routed* is started up, it reads the file */etc/gateways* to find gateways which may not be identified using the SIOGIFCONF *ioctl*. Gateways specified in this manner should be marked *passive* if they are not expected to exchange routing information, while gateways marked *active* should be willing to exchange routing information (i.e. they should have a *routed* process running on the machine). *Passive* gateways are maintained in the routing tables forever and information regarding their existence is included in any routing information transmitted. *Active* gateways are treated equally to network interfaces. Routing information is distributed to the gateway and if no routing information is received for a period of the time, the associated route is deleted.

The */etc/gateways* is comprised of a series of lines, each in the following format:

```
< net | host > name1 gateway name2 metric value < passive | active >
```

The *net* or *host* keyword indicates if the route is to a network or specific host.

*Name1* is the name of the destination network or host. This may be a symbolic name located in */etc/networks* or */etc/hosts*, or an Internet address specified in "dot" notation; see *inet(3N)*.

*Name2* is the name or address of the gateway to which messages should be forwarded.

*Value* is a metric indicating the hop count to the destination host or network.

The keyword *passive* or *active* indicates if the gateway should be treated as *passive* or *active* (as described above).

#### FILES

*/etc/gateways* for distant gateways

#### SEE ALSO

"Internet Transport Protocols", XSI 028112, Xerox System Integration Standard.  
*udp(5P)*

#### BUGS

The kernel's routing tables may not correspond to those of *routed* for short periods of time while processes utilizing existing routes exit; the only remedy for this is to place the routing process in the kernel.

*Routed* should listen to intelligent interfaces, such as an IMP, and to error protocols, such as ICMP, to gather more information.

## NAME

*rwhod* – system status server

## SYNOPSIS

*/etc/rwhod*

## DESCRIPTION

*Rwhod* is the server which maintains the database used by the *rwho*(1N) and *ruptime*(1N) programs. Its operation is predicated on the ability to *broadcast* messages on a network.

*Rwhod* operates as both a producer and consumer of status information. As a producer of information it periodically queries the state of the system and constructs status messages which are broadcast on a network. As a consumer of information, it listens for other *rwhod* servers' status messages, validating them, then recording them in a collection of files located in the directory */usr/spool/rwho*.

The *rwho* server transmits and receives messages at the port indicated in the "rwho" service specification, see *services*(4N). The messages sent and received, are of the form:

```

struct outmp {
    char    out_line[8];/* tty name */
    char    out_name[8];/* user id */
    long    out_time; /* time on */
};

struct whod {
    char    wd_vers;
    char    wd_type;
    char    wd_fill[2];
    int     wd_sendtime;
    int     wd_recvtime;
    char    wd_hostname[32];
    int     wd_loadav[3];
    int     wd_boottime;
    struct  whoent {
        struct outmp we_utmp;
        int    we_idle;
    } wd_we[1024 / sizeof (struct whoent)];
};

```

All fields are converted to network byte order prior to transmission. The load averages are as calculated by the *ruptime*(1N) program, and represent load averages over the 5, 10, and 15 minute intervals prior to a server's transmission. The host name included is that returned by the *gethostname*(2N) system call. The array at the end of the message contains information about the users logged in to the sending machine. This information includes the contents of the *utmp*(4) entry for each non-idle terminal line and a value indicating the time since a character was last received on the terminal line.

Messages received by the *rwho* server are discarded unless they originated at a *rwho* server's port. In addition, if the host's name, as specified in the message, contains any unprintable ASCII characters, the message is discarded. Valid messages received by *rwhod* are placed in files named *whod.hostname* in the directory */usr/spool/rwho*. These files contain only the most recent message, in the format described above.

Status messages are generated approximately once every 60 seconds. *Rwhod* performs an *nlist*(3C) on */unix* every 10 minutes to guard against the possibility that this file is not the system image currently operating.

**SEE ALSO**

*rwho*(1N), *ruptime*(1N).

**BUGS**

Should relay status information between networks. People often interpret the server dying as a machine going down.

**NAME**

telnetd - DARPA TELNET protocol server

**SYNOPSIS**

/etc/telnetd [-d] [port]

**DESCRIPTION**

*Telnetd* is a server which supports the DARPA standard TELNET virtual terminal protocol. The TELNET server operates at the port indicated in the "telnet" service description; see *services(4N)*. This port number may be overridden (for debugging purposes) by specifying a port number on the command line. If the **-d** option is specified, each socket created by *telnetd* will have debugging enabled (see *SO\_DEBUG* in *socket(2N)*).

*Telnetd* operates by allocating a pseudo-terminal device (see *pty(5)*) for a client, then creating a login process which has the slave side of the pseudo-terminal as *stdin*, *stdout*, and *stderr*. *Telnetd* manipulates the master side of the pseudo terminal, implementing the TELNET protocol and passing characters between the client and login process.

When a TELNET session is started up, *telnetd* sends a TELNET option to the client side indicating a willingness to do "remote echo" of characters. The pseudo terminal allocated to the client is configured to operate in "cooked" mode, (see *termio(7)*). Aside from this initial setup, the only mode changes *telnetd* will carry out are those required for echoing characters at the client side of the connection.

*Telnetd* supports binary mode, and most of the common TELNET options, but does not, for instance, support timing marks. Consult the source code for an exact list of which options are not implemented.

**SEE ALSO**

telnet(1N).



**NAME**

*ftpd* -- DARPA Trivial File Transfer Protocol server

**SYNOPSIS**

*/etc/ftpd [-d] [ port ]*

**DESCRIPTION**

*Tftpd* is a server which supports the DARPA Trivial File Transfer Protocol. The TFTP server operates at the port indicated in the "tftp" service description; see *services(4N)*. This port number may be overridden (for debugging purposes) by specifying a port number on the command line. If the *-d* option is specified, each socket created by *tftpd* will have debugging enabled (see *SO\_DEBUG* in *socket(2N)*).

The use of *tftp* does not require an account or password on the remote system. Due to the lack of authentication information, *tftpd* will allow only publicly readable files to be accessed. Note that this extends the concept of "public" to include all users on all hosts that can be reached through the network; this may not be appropriate on all systems, and its implications should be considered before enabling tftp service.

**BUGS**

This server is known only to be self consistent Due to the unreliability of the transport protocol (UDP) and the scarcity of TFTP implementations, it is uncertain whether it really works.

The search permissions of the directories leading to the files accessed are not checked.

**NAME**

*trpt* - transliterate protocol trace

**SYNOPSIS**

*trpt* [-a] [-s] [-t] [-j] [-phex-address] [system[core]]

**DESCRIPTION**

*Trpt* interrogates the buffer of TCP trace records created when a socket is marked for debugging (see *getsockopt(2N)*), and prints a readable description of these records. When no options are supplied, *trpt* prints all the trace records found in the system grouped according to TCP connection protocol control block (PCB). The following options may be used to alter this behavior.

- s in addition to the normal output, print a detailed description of the packet sequencing information,
- t in addition to the normal output, print the values for all timers at each point in the trace,
- j just give a list of the protocol control block addresses for which there are trace records,
- p show only trace records associated with the protocol control block who's address follows,
- a in addition to the normal output, print the values of the source and destination addresses for each packet recorded.

The recommended use of *trpt* is as follows. Isolate the problem and enable debugging on the socket(s) involved in the connection. Find the address of the protocol control blocks associated with the sockets using the -A option to *netstat(1N)*. Then run *trpt* with the -p option, supplying the associated protocol control block addresses. If there are many sockets using the debugging option, the -j option may be useful in checking to see if any trace records are present for the socket in question.

If debugging is being performed on a system or core file other than the default, the last two arguments may be used to supplant the defaults.

**FILES**

/unix  
/dev/kmem

**SEE ALSO**

*getsockopt(2N)*, *netstat(1N)*

**DIAGNOSTICS**

"no namelist" when the system image doesn't contain the proper symbols to find the trace buffer; others which should be self explanatory.

**BUGS**

Should also print the data for each input or output, but this is not saved in the trace record.

**Colophon**

**Composed at UniSoft Systems Inc.  
on the UniPlus<sup>+</sup> Operating System  
Designed by the Documentation Department  
Printed in Times Roman on Sequoia Matt**

